University of Alberta

SET REPRESENTATIONS OF GRAPHS

by

William Rosgen ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2005

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-494-09273-4
Our file Notre référence
ISBN: 0-494-09273-4

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

A set representation of a graph is an assignment of sets to the vertices of a graph in such a way that the sets assigned to two vertices satisfy a given relation if and only if the vertices are adjacent. The relations of intersection, overlap, and containment are studied, with emphasis on the number of elements required to form a representation.

The complexity of several combinatorial problems on the classes of graphs having small intersection representations that satisfy the Helly property, or equivalently, few maximal cliques, is studied. Also examined is the problem of extending set representations, with a proof of the computational hardness of extending overlap and containment representations. In addition, algorithms are given to find minimum intersection, overlap, and containment representations for some classes of graphs. Also presented are upper and lower bounds on the number of elements needed in these representations for graphs of various classes.

Acknowledgements

I would like to thank my supervisor, Lorna Stewart, who suggested this area of research, and pointed out several interesting research avenues. Without her insight, wisdom, and commitment for the last two years I would know substantially less about graphs. I would also like to thank Sean Kennedy, for listening to my bad ideas, as well as a few good ones, and generally being a helpful and considerate person to bounce ideas off of. Joe Culberson has also taught me a lot about research, including that occasionally you need to quantify everything and do some algebra. The bounds in Section 3.2.3 could easily have come from an assignment in his course. Joe has also had a few very helpful discussions with me about the nature of research, and my plans for the future. In addition, I would like to thank Piotr Rudnicki, for teaching me many techniques for finding efficient algorithms. The knowledge I gained from my experience on the programming team that he coached has been useful in a surprising number of ways. I am also grateful to Gerald Cliff, Ryan Hayward, and John Moon for reading an earlier version of this thesis and providing thoughtful suggestions on how to improve it.

Table of Contents

1 Introduction **1**
1.1 Overview 1
1.2 Preliminaries 1
1.2.1 Graph Classes 3
1.2.2 Computational Complexity 6
1.3 Set Representations 8
1.3.1 Representations by Restricted Families of Sets 10
1.4 Survey of Results 11

2 Intersection Representations **15**
2.1 Introduction 15
2.2 Hardness Results 17
2.3 Algorithms 21
2.3.1 Relation to the Edge Clique Cover Problem 22
2.3.2 Chordal Graphs and Related Classes 23
2.3.3 Comparability Graphs and Related Classes 25
2.3.4 Octahedral Graphs 26
2.3.5 Extending an Intersection Representation 29
2.4 Bounds on the Size of an Intersection Representation 31
2.5 Representations with the Helly Property 31
2.5.1 Characterizations of Graphs with Few Cliques 35
2.5.2 Graph Classes with Few Cliques 38
2.5.3 Complexity Results on Graphs with Few Cliques 39
2.5.3.1 Recognition, Robustness, and **NP**-completeness 40
2.5.3.2 Simple Complexity Results 41
2.5.3.3 k-Colourability 43
2.5.3.4 k-Clique Partition 44
2.5.3.5 Clique Partition and Clique Cover 46
2.5.3.6 Clique k-Partition 47

3 Overlap Representations **51**
3.1 Introduction 51
3.2 Algorithms 55
3.2.1 Cliques 55
3.2.2 Complete k-Partite Graphs 59
3.2.3 Computing a Representation of K_n 59
3.2.3.1 Bounds in the Size of the Graph 60
3.2.3.2 Calculating S(p,m) 63
3.2.3.3 Finding a Representation 65
3.2.4 Paths, Cycles, and Caterpillars 66
3.2.5 Connected Components 72
3.3 Hardness Results 75
3.3.1 Extending an Overlap Representation 76
3.3.2 Containment-Free Representations 80
3.3.3 Overlap Representations with Limited Containment 81
3.4 Bounds on the Size of an Overlap Representation 87

3.4.1 Relation to the Size of an Intersection Representation 89
3.4.2 Cocomparability Graphs 91

4 Containment Representations **93**
4.1 Introduction 93
4.2 Embedding a Partial Order into a Hypercube 96
4.2.1 Partial Order Dimension Theory 97
4.2.2 Complexity of Calculating the 2-Dimension 100
4.3 Hardness Results 101
4.3.1 Extending a Containment Representation 101
4.4 Bounds on the Size of a Containment Representation 104
4.4.1 Relation to Overlap Representations 105
4.5 Algorithms 106
4.5.1 Independent Sets 106
4.5.2 Complements of Paths and Caterpillars 108

5 Conclusions and Open Problems **111**

Bibliography **113**

A NP-complete Problems **117**

List of Tables

2.1 Complexity results on graphs with few cliques 33
3.1 Values of the function $S(p, m)$ 57

List of Figures

1.1 Containments between graph classes 4
1.2 Containments between complexity classes 7

2.1 Example of a minimum intersection representation 16
2.2 Example of the edge clique graph operator 22
2.3 Cograph with non-perfect edge clique graph 26
2.4 The graph W_4 26
2.5 The graph O_3 27
2.6 A graph whose complement has many cliques 35

3.1 Example of a minimum overlap representation 52
3.2 Example of a component in the reduction 83

4.1 Example of a minimum containment representation 95
4.2 Example showing that the 2-dimension is not a comparability invariant . . . 99

List of Symbols

$\overline{G}$	complement of graph G
$G[S]$	subgraph of G induced on S
$N(v)$	open neighbourhood of vertex v
$N[v]$	closed neighbourhood of vertex v
$\mathbb{N}$	set of natural numbers, $\{0, 1, 2, \ldots\}$
$\log n$	base 2 logarithm of n
$\|A\|$	size of set A
$A \setminus B$	set difference of A and B
P_n	path on n vertices
C_n	cycle on n vertices
K_n	clique on n vertices
O_n	complement of a perfect matching on $2n$ vertices
$d(u, v)$	shortest-path distance between u and v
$G \cup H$	disjoint union of graphs G and H
kG	disjoint union of k copies of G
$(H)_2$	2-section of hypergraph H
H^*	dual of hypergraph H
$K(G)$	edge-clique graph of G
$\theta(G)$	vertex-clique-cover number of G
$\alpha(G)$	size of a maximum independent set in G
$\theta_e(G)$	intersection number (edge-clique-cover number) of G
$\varphi(G)$	overlap number of G
$\xi(G)$	containment number of G
$\varphi'(G)$	disjointedness-free overlap number of G
$\xi'(G)$	disjointedness-free containment number of G
$\mathcal{K}(G)$	set of maximal cliques in G
$h(P)$	height of poset P
$\dim(P)$	dimension of poset P
$\dim_k(P)$	k-dimension of poset P

Chapter 1

Introduction

1.1 Overview

Representing the vertices of a graph by sets is a natural way to encode the adjacency information of the graph into local vertex labels. In many cases these representations can allow for both compact graph representations and efficient adjacency testing with only information stored locally with the vertices of the graph. If we seek such an efficient representation we are naturally concerned with the size of the representation, which is the problem we primarily study here.

The types of representations we concern ourselves with are those in which the vertices of a graph are assigned sets that have some property that determines whether the vertices of the graph are adjacent. As an example, we might insist that the sets assigned to vertices intersect if and only if the vertices are adjacent. This restriction forms an intersection representation, and we can form overlap and containment representations by requiring the sets of a representation to have other relations that determine the adjacency of the vertices in the represented graph. These are the three types of representation that have received the most attention in the literature and the three most obvious set relationships to consider, and so these are the three representations that we study here.

1.2 Preliminaries

In this section we present some notation that will be used throughout the thesis. A graph is an ordered pair of sets, typically written $G = (V, E)$, where the first set contains the vertices of the graph, and the second set, the edge set, contains unordered pairs of vertices. We will often use $n = |V|$ and $m = |E|$ to be the number of vertices and edges in a graph. We will say that two vertices $u, v \in V$ are *adjacent* if there exists a pair $(u, v) \in E$. We define the *degree* of a vertex to be the number of edges containing that vertex as the first element, or equivalently, the number of vertices that are adjacent to the vertex. $N(v)$ denotes the *open neighbourhood* of the vertex v, that is, the set of all vertices v is adjacent to. The *closed neighbourhood* is this set with the addition of the vertex v, and is denoted $N[v]$. In addition, all graphs we will consider are finite and simple, that is having no loops or multiple edges. Unless stated otherwise, the graphs we consider will be undirected, which is $(u, v) \in E$ if and only if $(v, u) \in E$, but we will on occasion consider the directed case. The complement of an undirected graph $G = (V, E)$, is the graph $\overline{G} = (V, \overline{E})$, where $\overline{E} = \{(u, v) : u, v \in V, (u, v) \notin E\}$.

Given a graph $G = (V_1, E_1)$, a graph $H = (V_2, E_2)$ is called a *subgraph* of G if $V_2 \subseteq V_1$ and $E_2 \subseteq E_1$, and further, H is an *induced* subgraph if $E_2 = \{(u, v) \in E_1 : u, v \in V_2\}$. If $S \subseteq V$ is a subset of the vertices of a graph $G = (V, E)$, then $G[S]$ denotes the subgraph of G induced on the vertices in S. A complete graph is a graph in which every pair of vertices is adjacent. We denote by the term *clique* a complete graph, or a complete induced subgraph when we refer to a clique of a graph G. A clique is *maximal* if it cannot be made into a larger clique with the addition of one more vertex of the graph. The clique on n vertices is denoted by K_n. We will occasionally use $\mathcal{K}(G)$ to denote the set of all maximal cliques in G. In addition to cliques, we can define an *independent set* to be a set of vertices such that no pair of them is adjacent, that is, the complement of a clique. We will use $\alpha(G)$ to denote the size of the maximum independent set in the graph G.

We refer to a vertex of a graph that is adjacent to all other vertices as a *universal vertex*. Similarly, a vertex is an *isolated vertex* if it is not adjacent to any other vertices. In a graph $G = (V, E)$, a set S of vertices that have identical neighbourhoods in $V \setminus S$ is called a *module*. In other words, a set of vertices is a module if for every vertex outside of the set, that vertex is either adjacent to all vertices of the set, or none of them.

We define a path as a graph with an ordered vertex set, such that each vertex is adjacent to the vertices that are immediately before or after it in the ordering. In addition, we require that the vertices on a path be distinct. A path on n vertices has exactly $n - 1$ edges. More formally, a path on n vertices is defined as $P_n = (\langle v_1, v_2, \ldots, v_n \rangle, \{(v_i, v_{i+1}) : 1 \leq i < n\})$. A cycle of length n, denoted C_n, is a path on n vertices, where we add an edge between v_1 and v_n.

We define the *distance* (or shortest path distance) between two vertices u and v as the length of a shortest path starting at u and ending at v. This is denoted by $d(u, v)$. By definition, we have $d(v, v) = 0$, for all vertices v. If there is no path between two vertices

u and v, then by definition $d(u, v) = \infty$.

A graph G is *connected* if there is a path between any two vertices, or equivalently, if there are no two vertices u and v such that $d(u, v) = \infty$. A graph is called *disconnected* if it is not connected, and the maximal connected subgraphs are referred to as the *components* of the graph.

For graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, with $V_1 \cap V_2 = \emptyset$, we define the *union* of G_1 and G_2 to be the graph formed by taking G_1 and G_2 without adding any connecting edges between them. More formally, we have $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. The union operator is a type of addition, hence we can make use of multiplicative notation to represent iterating this operator some natural number of times. To this end, for a graph G, and $t \in \mathbb{N}$ we can define, for $t > 1$, $tG = G \cup (t-1)G$, where $1G = G$.

A graph $G = (V, E)$ is covered by a set $\mathcal{C}$ of cliques if for each vertex in $v \in V$ there is some clique $S \in \mathcal{C}$ of G with $v \in S$. Such a set $\mathcal{C}$ is called a *clique cover*, or a *vertex clique cover*, of G, and the minimum number of cliques in any clique cover is the *clique covering number* of G, denoted $\theta(G)$. A similar concept is the edge clique cover, which is also a set $\mathcal{C}$ of cliques, but in this case we require, for every $(u, v) \in E$ that there is some clique $S \in \mathcal{C}$ of G such that $u, v \in S$. In other words, there must be some clique covering each edge of G. Such a set is called an *edge clique cover* of G, and the minimum number of cliques in any such cover, called the *edge clique cover number* of G, is denoted $\theta_e(G)$.

With these basic graph notions defined, we can begin introducing the classes of graphs that we will refer to, which is done in the next section.

1.2.1 Graph Classes

In this section we examine some of the classes of graphs that we will encounter later. This survey is by no means complete, and the interested reader is referred to the book of Brandstädt, Le, and Spinrad [5] that includes more information on all of the classes of graphs discussed here. We have already seen, in the previous section, some simple classes of graphs, such as paths, cycles, cliques, and independent sets. Many of the known containment relationships for the graph classes discussed here are shown in Figure 1.1.

The first class of graphs we introduce is also among the simplest. A graph is a *tree* if it is connected and acyclic. The *leaves* of a tree are those vertices with degree one. A tree is *rooted* if there is a specific node, the *root* of the tree, that specifies an orientation of the tree, with all edges directed away from the root. In such a rooted tree, the *children* of a node are those nodes it is adjacent to via edges directed away from the node, and the *parent* of

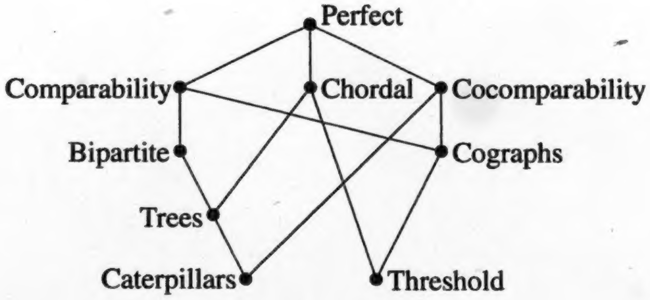

Figure 1.1: Containments between the graph classes discussed here.

the node is the node that can be reached by travelling up an edge in the opposite direction to its orientation. The root node has no parent, but all other nodes have a unique parent. Similarly, the leaves of a tree have no children, but all other nodes do. A forest is a graph whose connected components all form trees, that is, an acyclic graph that is not necessarily connected.

A subclass of trees are the caterpillars. Caterpillars are formed from paths by adding leaves. More formally, a *caterpillar* is a graph consisting of degree one vertices and a path. This path is called the *spine* of the caterpillar. It is of note that caterpillars are cocomparability graphs, but trees are not, where the cocomparability graphs are a class of graphs we will encounter later in this section.

The *bipartite* graphs are those graphs that do not contain odd cycles, including the three cycle K_3. This causes them to contain the class of trees. This definition is equivalent to the usual definition, in which a graph $G = (V, E)$ is bipartite if there is a partition of V into U and W such that every edge in E has one endpoint in each of U and W. This class of graphs is interesting since it does not contain K_3, which also forbids any cliques larger than K_3.

Another class of graphs with limits on the types of induced cycles is the class of *chordal graphs*. These are the graphs that do not contain induced cycles of length more than three. In other words, a graph is chordal if it does not contain C_k for any $k \geq 4$. One interesting property of the chordal graphs is that the vertices can be ordered in such a way that the neighbours of any vertex that appear after the vertex in the ordering form a clique. Such an ordering is known as a *perfect elimination ordering*, and the chordal graphs can be characterized as exactly those graphs that have perfect elimination orderings.

The *threshold graphs* are also a class of graphs that can be characterized by an ordering on the vertices of the graph. In this case, the vertices are ordered in such a way that when they are added, in order, to the empty graph, with a restriction on the edges added, the original threshold graph is obtained. The restriction on the added edges is that when adding a new vertex to the graph it must either be a universal vertex or an isolated vertex. This restriction can make problems easy to solve on threshold graphs, as local operations can often be used to maintain the optimality of a solution through the addition of a new vertex.

The graphs without P_4 as an induced subgraph are known as the *cographs*. This characterization, while simple, ignores the feature of cographs that is most often exploited algorithmically. This feature is the fact that any cograph with at least two vertices can be decomposed into two cographs that are either disjoint or completely connected to each other. This structure permits many combinatorial problems to be solved by divide and conquer algorithms on this class of graphs.

The cographs are contained in both the comparability graphs and the cocomparability graphs. The *comparability* graphs are the graphs whose edges have a transitive orientation, where the notion of a transitive orientation is given by the following definition.

Definition 1.1. Given an undirected graph $G = (V, E)$, a directed graph $H = (V, E')$ is a *transitive orientation* of G if $E' \subseteq E$ and the following two conditions are satisfied

1. If $\{(u,v),(v,u)\} \subseteq E$ then $|\{(u,v),(v,u)\} \cap E'| = 1$
2. If $(u,v) \in E'$ and $(v,w) \in E'$ then $(u,w) \in E'$

The first of these two conditions ensures that H is an *orientation* of G, which is, each undirected edge of G is oriented in exactly one of two possible ways. The second condition ensures that this orientation is transitive.

Since we have defined the comparability graphs, we can define the *cocomparability graphs*. These graphs are exactly the complements of the comparability graphs, that is, the graphs whose non-edges can be transitively oriented. We will encounter these classes of graphs when studying containment representations, since a graph has a containment representation if and only if it is a comparability graph.

An important class of graphs that contains the comparability graphs, the cocomparability graphs, and the chordal graphs is the class of perfect graphs. A graph G is *perfect* if for any induced subgraph G' of G the size of the maximum clique in G' is equal to the minimum number of colours in any colouring of G', where a *colouring* of G' is an assignment of colours to the vertices of G' such that no two adjacent vertices are assigned the same colour.

An excellent survey of some of the more famous classes of perfect graphs, including many of those that we have discussed, is the 1984 survey by Duchet [13].

One other class of graphs that we will use are the *planar graphs*. The simple description of this class is that these are the graphs that can be embedded on a plane in such a way that no two edges cross. One graph that is not planar is the graph K_5, and we will see later that this limits the number of maximal cliques in any planar graph, which will be the reason that we consider this class of graphs.

These are not all of the classes of graphs that we will consider, but we delay the introduction of the classes of graphs defined in terms of set representations until Section 1.3.1, which will allow us to formally introduce the notions of intersection, overlap, and containment representations of graphs.

1.2.2 Computational Complexity

This section briefly discusses the notions that we will use from the theory of computational complexity.

The class **P** contains all those problems solvable by a deterministic algorithm in time polynomial in the size of the input. This class contains all of the "easy" problems, or problems for which there exist efficient algorithms that find solutions. A superclass of **P** is **NP**, which informally contains those problems for which solutions can be efficiently verified. More formally, **NP** consists of those problems that can be solved with a nondeterministic algorithm that can guess the solution to a problem, and then verify, in polynomial time, that the solution is correct. Another way to view this is that any problem in **NP** is a problem in **P** with an extra existential quantifier that asserts there is some guessable solution that a deterministic polynomial time algorithm can solve.

The hardest problems in **NP** are the **NP**-complete problems which can be solved by a polynomial time algorithm if and only if all other problems in **NP** can. The problem of finding a satisfying truth assignment for a boolean formula is the canonical **NP**-complete problem. An exposition of these concepts, as well as some of the hardness results used later, can be found in [21].

The remainder of this section is a quick and informal discussion of some of the larger complexity classes that we will encounter. For a more thorough discussion of these classes see [41]. The known containment relationships between the complexity classes we will encounter here are given as Figure 1.2. None of the containments listed here are known to be proper, but it is believed that many of them are.

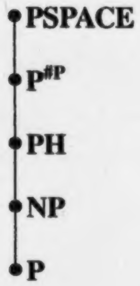

Figure 1.2: Containments between the complexity classes discussed here.

We will encounter problems that appear to be even harder than the **NP**-complete problems. These problems are counting problems, such as counting the number of valid 3-colourings of a given graph. The class of such problems, introduced in 1979 by Valiant [63], is known as **#P**. More formally, **#P** is the class of functions that count the number of solutions to combinatorial problems. Since **#P** is a class of functions, and other complexity classes contain problems, which are represented as sets, the class **#P** cannot appear in Figure 1.2, except as $\mathbf{P}^{\#\mathbf{P}}$ which, as we discuss later, is the class of problems solvable in polynomial time with the ability to call to a function in **#P**. Fortunately we will be able to simply view **#P** as the class containing all counting problems associated with some relation that can be decided in polynomial time. The canonical **#P**-complete problem is that of computing the number of truth assignments that satisfy a boolean formula.

In order to understand the size of the class **#P**, we must introduce another group of complexity classes: the polynomial hierarchy. The polynomial hierarchy, introduced in 1972 by Meyer and Stockmeyer [38], is an extension of the class **NP**. Recall that **NP** can be thought of those problems is **P** with an added existential quantifier. Informally, the polynomial hierarchy is constructed of levels, where the kth level of the hierarchy consists of those problems in **P** with the addition of k nested quantifiers. It is believed that this hierarchy is infinite, with each level properly containing the previous levels, but proving such a result would be a major breakthrough in complexity theory. The class **PH** denotes the union of all levels of this hierarchy.

The complexity classes we consider are bounded by the class of problems that can be computed in a polynomially bounded amount of space, using potentially an exponential amount of time. The class of all such problems is known as **PSPACE**, and is only important here as an upper bound on the complexity of some of the problems we consider. This class is known to contain **PH** [54] and any function in **#P** can be computed in **PSPACE** [63] as well.

An important theorem in complexity theory, known as Toda's Theorem [57], states that **PH** is contained in a complexity class that is closely related to those problems that can be solved by a deterministic polynomial time algorithm with access to an oracle that can compute a function in **#P**. The class of problems solvable by such an algorithm is denoted $\mathbf{P}^{\#\mathbf{P}}$. A further consequence of this theorem is that there is such an algorithm that makes only one call to the function in **#P**. This seems to indicate that the class **#P** is much larger than the class **NP**, in terms of computational difficulty.

1.3 Set Representations

In order to introduce the different types of set representations of graphs that we will examine, we need to introduce the relationships between sets that will be used to build these representations.

Definition 1.2. For two sets A and B, we define the following set relationships:

- A and B *intersect* if $A \cap B \neq \emptyset$,
- A and B *overlap* if $A \cap B \neq \emptyset$ and neither $A \subseteq B$ nor $B \subseteq A$,
- A and B have a *containment* relationship if $A \subseteq B$ or $B \subseteq A$.

We will refer to a set of sets as a *family*, and we will use *collection* to refer to a multiset of sets. When forming set representations of graph we usually allow the same set to be assigned to multiple vertices, and thus it is natural to consider the representation as a collection.

The next definition is a property that we will apply to set representations of graphs. It is called the *Helly* property, as it is similar to Helly's Theorem that, as described in [65], states that for any family of sets in $\mathbb{R}^d$ such that any $d+1$ have nonempty intersection, the intersection of the entire family must also be nonempty. In the definition of the related property, the attachment to $\mathbb{R}^d$ is relaxed and we consider only those subfamilies that have pairwise nonempty intersection. A definition of this property follows.

Definition 1.3. A collection of sets satisfies the *Helly property* if for any subcollection that intersects pairwise, there is a common element in all sets of the subcollection.

Given this property and the set relationships of Definition 1.2, we can introduce the notion of a set representation for a graph. There are three major representations we will concern ourselves with, mirroring the three set relationships introduced earlier. These are intersection, overlap, and containment representations. These three forms of representation are special cases of the following general definition.

Definition 1.4. Given a graph $G = (V, E)$, a collection $\mathcal{C} = \{S_v : v \in V\}$, and a relation $R \subseteq \mathcal{C} \times \mathcal{C}$, the collection $\mathcal{C}$ forms an *R-representation* of G if

$$(S_u, S_v) \in R \text{ if and only if } (u, v) \in E.$$

If, in addition, $\mathcal{C}$ has the Helly property, we say that $\mathcal{C}$ is a *Helly R-representation* for G.

From this definition, we can immediately define intersection, overlap, and containment representations by using one of the relations defined in Definition 1.2 in place of R. Note that since we have defined containment relationship to be the symmetric closure of the usual containment representation, any graph having intersection, overlap, or containment representation is necessarily undirected, as each of these set relationships, in the context of a representation for a graph, are defined in a symmetric manner.

It should also be noted that when specifying an R-representation $\mathcal{C}$ for a graph $G = (V, E)$ we implicitly associate with the vertex v the set S_v. This is done as a notational convenience, as it allows us to avoid the introduction of a specific mapping from V to $\mathcal{C}$. This does not imply that the sets associated with different vertices are distinct. As $\mathcal{C}$ is a collection, we may have, for $u \neq v$, $S_u = S_v$, as we consider this set, in the collection, to have multiplicity at least two.

The parameter of an R-representation that we will be most concerned with is the size of a representation, or more precisely, we will study the size of a minimum representation for a given graph.

Definition 1.5. Let $G = (V, E)$ be a graph, and $\mathcal{C}$ an R-representation of G. The *size* of $\mathcal{C}$ is given by

$$\left| \bigcup_{S_v \in \mathcal{C}} S_v \right|.$$

The minimum size of an R-representation of a graph G is called the *R-number* of G. The *Helly R-number* of a graph G is the minimum size of a Helly R-representation of G.

Notice that from this definition, if we have an R-representation for G, then we can form an R-representation for an induced subgraph of G by simply taking the sets of the R-representation that correspond to the vertices of the subgraph. This implies that any induced subgraph of G has R-number no larger than the R-number of G.

Once again, the specific cases that we will examine are the intersection number, overlap number, and containment number. Since we will repeatedly encounter these graph parameters we introduce notation for them.

Definition 1.6. Let G be a graph. The *intersection number* of G is denoted by $\theta_e(G)$, the *overlap number* of G is given by $\varphi(G)$, and the *containment number* of G is denoted by $\xi(G)$.

It is these three parameters, and the computation of them, that is the primary focus of this thesis. It is no coincidence that the intersection number and the edge clique cover number of a graph are denoted by the same symbol, since we will see in Chapter 2, where we study intersection representations, that these two parameters are in fact equal. Overlap representations of general graphs and specific classes of graphs will be examined in Chapter 3. As will be seen in Chapter 4, where we study containment representations, not every graph has a containment representation, and so, if a graph G does not have such a representation we define $\xi(G) = \infty$. Every graph does, however, have overlap and intersection representations.

1.3.1 Representations by Restricted Families of Sets

In this section we introduce some classes of graphs that are defined in terms of intersection and overlap representations of restricted families of sets. In Definition 1.4 we allowed the sets of a representation to be arbitrary, but here we will require that all sets of a representation are drawn from some family of sets.

The first and arguably most famous class of graphs that we will consider are the *interval graphs*. These are the intersection graphs of intervals on a line. This is another class of graphs for which we will be able to find a bound on the number of maximal cliques.

Generalizing interval graphs are the *boxicity k graphs*, introduced by Roberts [48], which are the intersection graphs of k-dimensional "boxes" that are required to be axis-aligned. These objects are a natural generalization of intervals to k-dimensions, and by definition, the interval graphs are exactly the graphs of boxicity 1.

Another generalization of interval graphs are the *circular-arc graphs*. These are the intersection graphs of arcs on a circle. Clearly an interval graph can be represented as arcs of a circle, but there exist circular-arc graphs, such as cycles of length at least four, that cannot be represented as intersection graphs of intervals.

Related to the circular-arc graphs are the *circle graphs*. These graphs are usually defined as the intersection graphs of chords of a circle. By observing that two chords intersect if and only if the arcs they subtend overlap (choosing an arbitrary arc for those chords that cross the center of the circle), these graphs are exactly the graphs that can be represented as overlap graphs of arcs on a circle.

One class of graphs that we have already seen, the chordal graphs, can also be defined in terms of intersecting objects of a restricted family of sets. This is the result of Gavril [22] that shows that a graph is chordal if and only if it is the intersection graph of subtrees of some tree, where subtrees are taken to be sets of vertices that induce subtrees of the given tree. This is an interesting result that demonstrates the power of the idea of set representations for graphs, as a class of graphs defined in terms of forbidden subgraphs turns out to be exactly a class of graphs defined in terms of set intersection.

1.4 Survey of Results

In this section we survey the original contributions of this thesis, in approximately the order that they appear.

The first of these contributions is the determination of the intersection number for the *octahedral graphs*. These graphs are the complements of *perfect matchings*, which in turn are the graphs consisting of a collection of disjoint edges. This result is shown by connecting any intersection representation for such a graph to an overlap representation for a clique. These results appear in Section 2.3.4. Due to the connection shown between intersection representations of these graphs and overlap representations for cliques, this problem can be solved with the application of a theorem from combinatorics that can also be used to find the size of a minimum overlap representation for a clique, which is done in Section 3.2.1. Any minimum overlap representation for a clique can be converted into a minimum overlap representation for a complete k-partite graph. This conversion is demonstrated in Section 3.2.2. The bounds on the size of these representations are then converted into an algorithm by finding asymptotically tight bounds on the quantities involved, which is done in Section 3.2.3.

Another contribution is the study of the problem of extending a representation. Informally, this is the problem of, given a representation for a graph, can we extend this representation to include an extra vertex without increasing the size of the representation? In the case of intersection representations, there is an efficient algorithm to decide this problem, which is discussed in Section 2.3.5. In contrast, as discussed in Sections 3.3.1 and 4.3.1, the corresponding problems for overlap and containment representations are **NP**-complete.

These results lend evidence to the hypothesis that computing overlap and containment representations are at least as difficult as computing intersection representations.

In Section 2.5 the number of maximal cliques in a graph is shown to be the size of a minimum Helly intersection representation. This fact is not new, as it can be shown via a classical theorem on hypergraphs. The hardness of various combinatorial problems, given as input a Helly intersection representation, is, however, something that has not been studied before. By the result on hypergraphs, this is exactly the study of these combinatorial problems on graphs with a polynomially bounded number of maximal cliques, which have been introduced by Prisner [43], but a detailed analysis of the hardness of various problems on these graphs is a contribution of this thesis.

Computing overlap representations of various types of graphs is considered in Section 3.2. The problem of finding the overlap number for a graph in most classes of graphs remains open, but we are able to find overlap numbers for cliques, paths, cycles, and caterpillars. The primary difficulty in showing the optimality of these representations is showing a lower bound on the size that any overlap representation must have, as once we have shown a tight lower bound, it is usually trivial to construct a representation that achieves the bound. We also present, in this section, a constructive proof that can be seen as an efficient algorithm for a method of finding a minimum overlap representation for a disconnected graph, given minimum overlap representations for each component of the graph.

In addition to these algorithms for computing overlap representations, we are able to show the hardness of the problem of finding an overlap representation that is allowed to have only k containment relationships among the nonadjacent vertices, for any nonnegative constant k. This lends further evidence to the conjecture that computing the overlap number is computationally difficult. These results appear in Section 3.3.

There are upper bounds known for the intersection number of any graph, and in Sections 3.4 and 4.4 we consider upper bounds for the overlap and containment numbers, respectively. We observe that for the cocomparability graphs, and some classes of graphs contained in them, the size of the overlap and containment numbers must be linear in the number of vertices of the graph, while there are graphs in these classes that have intersection number that is quadratic in the number of vertices of the graph. We are also able to extend the upper bound on the intersection number given by the number of maximal cliques in a graph, with only an additive increase in the upper bound. This technique produces a linear upper bound on the overlap number of any planar graph.

In addition to these bounds, we show, in Section 4.4.1, a relationship between the overlap number of a graph and the containment number of the complement graph. Using this bound we are able to find lower bounds on the containment number for the complements of paths and caterpillars. Armed with this lower bound we need only to construct a representation with the correct size, which we are once again able to do.

No Text

Chapter 2

Intersection Representations

2.1 Introduction

Intersection representations are the set representations for graphs that have received the most study, although most of this study has been focused on intersection representations made up of sets from some restricted family, such as intervals on a line or subtrees of a tree. To formalize the notion of an intersection representation, we use the following definition.

Definition 2.1. Given a graph $G = (V, E)$, a collection $\mathcal{C} = \{S_v : v \in V\}$ is an *intersection representation* for G if for every $u, v \in V$ we have

$$(u, v) \in E \text{ if and only if } S_u \cap S_v \neq \emptyset.$$

We define the size of a representation as the number of elements in the union of all sets in the collection, which is

$$\left| \bigcup_{v \in V} S_v \right|,$$

and we let the *intersection number*, $\theta_e(G)$, be the size of a minimum intersection representation for the graph G.

One feature of this definition is that we allow an intersection representation to assign the same set to multiple vertices. This will allow for compact representations of vertices that are both cliques and modules. In other words, given any two adjacent vertices that have identical adjacency with the rest of the graph, we can always assign the same set to both vertices. As an example, the graph in Figure 2.1 has a minimum intersection representation with different vertices assigned the same set. Stated another way, the intersection number of a graph does not increase under *vertex expansion*, which replaces a vertex by two connected vertices that are adjacent to exactly those vertices that the original vertex was. This is summarized by the following lemma.

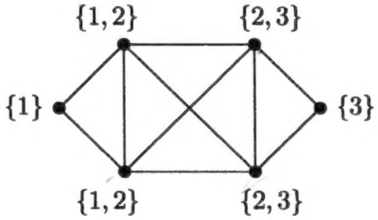

Figure 2.1: Example of a minimum intersection representation.

Lemma 2.2. *If H can be obtained from G by vertex expansion, then* $\theta_e(G) = \theta_e(H)$.

Proof. Since G is an induced subgraph of H, we immediate have $\theta_e(G) \leq \theta_e(H)$, since we can simply restrict a representation for H to the vertices of G.

In the other direction, let $\mathcal{C} = \{S_v : v \in V\}$ be an intersection representation for G, and let A_v be the set of all vertices in H that a vertex v of G is expanded into. If we assign the set S_v to all vertices in A_v, then two vertices u, w of H are adjacent in the representation if and only if they are in the same set A_v for some v, or if they are in sets A_v and A_z such that $S_v \cap S_z \neq \emptyset$. This second condition is exactly the adjacency condition for G, so the constructed representation is a valid intersection representation for H, which proves that $\theta_e(G) = \theta_e(H)$. □

The notation, $\theta_e(G)$, that we use for the intersection number is also notation that is commonly used for the minimum number of cliques needed to cover the edges of the graph. This is no accident, as we will see in Section 2.3.1 that the intersection number is exactly the edge clique cover number. This equivalence often allows the intersection number to be easily computed. We will also see in Section 2.3.1 a method to find the edge clique cover by forming a vertex clique cover of a different graph. Some of the known algorithms to find the intersection number exploit the fact that for many classes of graphs, a vertex clique cover for the graph that is constructed by this technique can be found in polynomial time.

Like overlap representations, but unlike containment representations, every graph has an intersection representation. This was noted in 1945 by Marczewski [56]. One way to see this is to notice that if to each vertex we assign the set of all edges incident to it, then two vertices will have intersecting sets if and only if they are endpoints of some edge. This representation is quite large, but we will see results in Section 2.4 that show that the intersection number is often required to be quadratic in the number of vertices of the graph.

The classes of graphs that are intersection graphs of some family of sets are characterized in [50]. Here it is shown that any class of graphs that is hereditary, closed under vertex expansion, and has a composition sequence is exactly the class of graphs that have intersection representations consisting of sets of some family. A class of graphs is *hereditary* if any induced subgraph of a graph in the class remains in the class. The second condition appears because any graph with an intersection representation as sets from a family we can form a similar representation, using the technique of Lemma 2.2, for any graph that can be reached by vertex expansion. The final condition in the characterization is more esoteric. A *composition sequence* for a class of graphs is a sequence of graphs $G_1, G_2, \ldots$ such that each graph in the sequence is in the class, each graph G_i is an induced subgraph of G_{i+1}, and for any graph G in the class, there is some i such that G is an induced subgraph of G_i. Many of the properties of intersection graphs can be seen in these conditions.

The remainder of this chapter is organized as follows. In Section 2.2 we examine some of the problems related to intersection representations that are known to be **NP**-complete. We follow this with a discussion, in Section 2.3, of the problems on intersection representations that are known to have polynomial time algorithms. We then examine, in Section 2.4, some upper bounds on the size of intersection representations. The case of Helly intersection representations is studied in Section 2.5, where we tie the size of a minimum Helly intersection representation to the number of maximal cliques in a graph. We also examine, in this section, the complexity of several problems where we are given a Helly intersection representation, or equivalently, where the number of maximal cliques in the input graph is bounded by some polynomial in the input size.

2.2 Hardness Results

In this section we examine the known hardness results related to the problem of finding a minimum intersection representation. Finding such a representation is closely related to the problem of covering a graph with cliques. As we have seen in the previous section, every graph has an intersection representation, as we can simply take as the set for each vertex all edges incident to it. A natural question to ask, given a graph, is: how many elements are required to form an intersection representation? A formalization of this is given by the following problem.

Problem. The INTERSECTION NUMBER problem is defined as:

Instance: A graph, $G = (V, E)$, and an integer k.

Question: Is there an intersection representation $C = \{S_v : v \in V\}$ of G such that

$$\left|\bigcup_{v \in V} S_v\right| \leq k?$$

We will defer the discussion of the hardness of this problem until after we have introduced some other hard problems related to it, as the number of elements in the union of the representation is not the only measure of size that we might consider. One other option for the size measure is given by what we refer to here as the UNIFORM INTERSECTION NUMBER problem, where we seek the minimum k such that we can form an intersection representation using only sets of size k. This notion is also studied in relation to hypergraphs [44], where a graph has uniform intersection number at most k if and only if it can be represented as the intersection graph of a k-uniform hypergraph that may have duplicate edges, which is simply a collection of sets of size k over some set of elements. This problem is equivalent to the variant of the problem where we allow the sets in the representation to have size at most k as defined in [42]. These problems are clearly equivalent, as we can add new elements to any sets that are not of the correct size to transform one representation to the other. A formalization of the problem follows.

Problem. The UNIFORM INTERSECTION NUMBER problem is defined as:

Instance: A graph, $G = (V, E)$, and an integer k.

Question: Is there an intersection representation $C = \{S_v : v \in V\}$ of G with, for all $v \in V$,

$$|S_v| \leq k?$$

We can also define the k-UNIFORM INTERSECTION NUMBER problem, which is simply the UNIFORM INTERSECTION NUMBER problem, with a fixed value of k. In the literature this problem is often referred to as the k-INTERSECTION NUMBER problem, but we add "Uniform" to the name of the problem in order to avoid ambiguity with the problem of finding a p-intersection representation where the vertices associated with two sets are considered adjacent only if the size of their intersection is at least p.

Problem. The k-UNIFORM INTERSECTION NUMBER problem is defined as:

Instance: A graph, $G = (V, E)$.

Question: Is there is an intersection representation $\mathcal{C} = \{S_v : v \in V\}$ of G with, for all $v \in V$,

$$|S_v| \leq k?$$

In 1981 these two problems were shown to be **NP**-complete by Poljak, Rödl, and Turzík [42]. These hardness results appear as the following theorems.

Theorem 2.3 (Poljak, Rödl, and Turzík [42]). *The* UNIFORM INTERSECTION NUMBER *problem is* **NP**-*complete.*

The hardness of k-UNIFORM INTERSECTION NUMBER is only shown for $k = 4$ in [42], but by a simple application of one of the techniques used in the paper, the addition of pendant vertices attached to all vertices in the graph, we can easily reduce k-UNIFORM INTERSECTION NUMBER to $(k+1)$-UNIFORM INTERSECTION NUMBER. To see this, we create, from the graph $G = (V, E)$, the graph $G' = (V \cup V', E \cup E')$, where $V' = \{v' : v \in V\}$ and $E' = \{(v, v') : v \in V\}$; that is, we copy each vertex, and make the copy adjacent only to the original vertex. The claim is that the uniform intersection number of G' is one more than that of G. In one direction, given an intersection representation $\mathcal{C} = \{S_v : v \in V\}$ of sets having size at most k for G, we can form the representation $\mathcal{C}' = \{S'_v = S_v \cup \{v\} : v \in V\} \cup \{S'_{v'} = \{v\} : v' \in V'\}$. Notice that $\mathcal{C}'$ preserves the representation on G, and correctly represents the new edges in E' as well. In the other direction, let $\mathcal{C}'$ be a $k+1$-uniform intersection representation for G'. Let $v' \in V'$, and notice that the set $S'_{v'}$ associated with $v' \in V'$ must be disjoint from all sets of the representation, with the exception of S'_v. Thus, since $S'_{v'}$ is nonempty, if we remove it, and the elements in it from S'_v, we obtain a representation for the graph G' with the vertex v' removed, and the set S'_v is at least one smaller. Doing this for all vertices, we construct the representation $\mathcal{C} = \{S_v = S'_v \setminus S'_{v'} : v \in V\}$ for the graph G, and for all v we have $|S_v| \leq k$, since there was at least one element in the intersection of S'_v and $S'_{v'}$. Thus, the uniform intersection number of G' is one more than the uniform intersection number of G. Implementing this transformation as a reduction, noting once more that this technique was given in [42] but not applied to this theorem, we can extend the hardness of k-UNIFORM INTERSECTION NUMBER to all $k \geq 4$.

Theorem 2.4 (Poljak, Rödl, and Turzík [42]). *For any $k \geq 4$, the k-*UNIFORM INTERSECTION NUMBER *problem is* **NP**-*complete.*

Also in [42] are proofs of the hardness of some similar problems, such as the UNIFORM INTERSECTION NUMBER problem where the sets are required to be distinct, or more strictly, where any two sets can have intersection of size at most one. Even deciding whether the uniform intersection number is equal to the uniform intersection number when we require the sets to be distinct is **NP**-hard, even though we can transform any uniform representation to a distinct one with sets at most one larger by simply adding a new element to each set.

Having discussed a few variants of the problem, we are now prepared to present the hardness result of Kou, Stockmeyer, and Wong [34], which shows that the INTERSECTION NUMBER problem is **NP**-complete. The first step of this result is showing that the problem of finding a minimum edge clique cover is equivalent to the problem of finding a minimum intersection representation. This forms the following theorem.

Theorem 2.5 (Kou, Stockmeyer, and Wong [34]). *For any graph G, the intersection number of G is equal to $\theta_e(G)$, the edge clique cover number of G.*

Proof. Take any edge clique cover $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ of $G = (V, E)$ having size k. Consider the collection given by

$$\mathcal{D} = \{S_v = \{C \in \mathcal{C} : v \in C\} : v \in V\}.$$

If two vertices $u, v \in V$ are adjacent, then the edge between them will be covered by some $C \in \mathcal{C}$, and thus, in the representation we have $S_u \cap S_v \neq \emptyset$. On the other hand, if $u, v \in V$ are not adjacent, they cannot be in the same clique, and so S_u and S_v will be disjoint. Thus $\mathcal{D}$ is an intersection representation for G using k elements.

In the other direction, let $\mathcal{D} = \{S_v : v \in V\}$ be an intersection representation with k elements, given by $\{1, 2, \dots, k\}$. Consider the collection given by, for each $1 \leq i \leq k$,

$$\mathcal{C} = \{\{v : i \in S_v\} : 1 \leq i \leq k\}.$$

Each set in the collection represents those vertices whose set representation has a specific element i. Thus, these vertices must all be adjacent and $\mathcal{C}$ is a collection of cliques in G. In addition, for every edge $(u, v) \in E$ there is some element $i \in S_u \cap S_v$, and so there is a clique in $\mathcal{C}$ that covers this edge. Thus there is an edge clique cover using k cliques, and so we have shown that a minimum edge clique cover is equivalent to a minimum intersection representation. □

Also, notice that the transformations used in the proof of this proposition can be implemented in time linear in the size of a description of the representation. That is, this can be done in time linear in the sum of the sizes of all sets in the representation, or the sum of the sizes of all cliques in an edge clique cover. This fact will be useful in Section 2.3 when we consider algorithms to solve the intersection number problem on restricted classes of graphs.

It is also shown in [34] that any instance of the vertex clique cover problem can be transformed into an instance of the edge clique cover problem, by adding $m + 1$ additional universal vertices, where m is the number of edges in the graph. With the **NP**-completeness of the vertex clique cover problem [32] and the previous proposition, this implies the **NP**-completeness of the INTERSECTION NUMBER problem.

Theorem 2.6 (Kou, Stockmeyer, and Wong [34]). *The* INTERSECTION NUMBER *problem is* **NP**-*complete.*

2.3 Algorithms

The problem of finding a minimum intersection representation for a general graph, as we have seen in Section 2.2, is **NP**-complete. There are, however, classes of graphs on which we can find efficient algorithms to generate minimum intersection representations. In this section, we examine algorithmic results on some of these classes of graphs.

One immediate example of a class of graphs for which we can efficiently find a minimum intersection representation are the triangle free graphs. A minimum edge clique cover for a triangle free graph is given by the collection of all edges in the graph, as each edge is contained in exactly one clique. Thus, if we take the set of all edges incident on each vertex as the sets of the representation, we obtain a minimum intersection representation for any triangle free graph. This can be performed in linear time.

Before examining any further algorithms for finding the intersection number, we will need to introduce the concept of an edge clique graph, which we do in Section 2.3.1. This concept will allow us to relate the problem of finding an edge clique cover to the problem of finding a vertex clique cover. We will see in Section 2.3.2 an algorithm that, using the results in Section 2.3.1, can be applied to the class of chordal graphs, which contains the class of triangle free graphs discussed above. We will also see, in Section 2.3.3, an algorithm to find a minimum intersection representation for a subclass of the comparability graphs, and we will examine, in Section 2.3.4, a method to find a minimum intersection representation

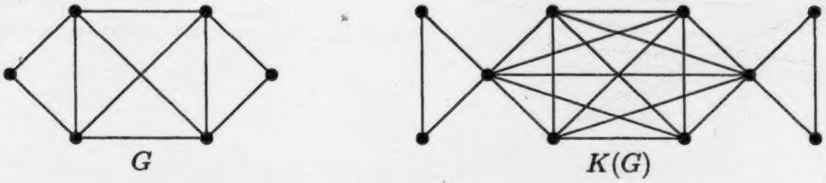

Figure 2.2: A graph and the edge clique graph obtained from it.

for a specific class of comparability graphs that are not in covered by the previous result. The study of intersection representations for this class of graphs will lead to the study of overlap representations on cliques, which is a discussion we defer to Section 3.2.1. Finally, we will examine a polynomial-time algorithm to decide if an intersection representation can be extended to add an additional vertex without adding additional elements to the representation. This is in contrast to similar problems for overlap and containment representations, which are **NP**-complete, as we will show in Section 3.3.1 and Section 4.3.1.

2.3.1 Relation to the Edge Clique Cover Problem

An important building block in an algorithm to find the intersection number of a graph is the concept of an edge clique graph. This concept appears implicitly in the proof of the hardness results in [34], and it was first studied explicitly in [1]. Edge clique graphs have since received considerable study, both in terms of characterizations [10, 12], and in terms of edge clique graphs of restricted classes of graphs [6, 11, 45, 46].

Definition 2.7. For a graph $G = (V, E)$, the *edge clique graph* of G is denoted $K(G)$, and is given by the graph $K(G) = (E, F)$, where for any distinct $e_1, e_2 \in E$ we have $(e_1, e_2) \in F$ if and only if the vertices that make up the endpoints of e_1 and e_2 form a clique in G.

Notice that the vertices of the edges e_1 and e_2 need not be distinct. As an example, a graph and its edge clique graph are shown in Figure 2.2. We can also view K as an operator on graphs. The class of *edge clique graphs*, which are those graphs G for which there exists a graph H such that $K(H) = G$, are characterized in [10], but it remains unknown if this class can be recognized in polynomial time.

Having defined the edge clique graph operator, we motivate it with the following proposition, shown in [34], which we will make use of when considering algorithms to find the intersection number of a graph.

Proposition 2.8 ([34]). *For a graph G, the size of a minimum edge clique cover of G is equal to the size of a minimum vertex clique cover of $K(G)$.*

Proof. Take any vertex clique cover of $K(G)$. Each clique in this cover forms a collection of edges in G, each pair of which is in a common clique of G. This implies that these edges are contained in some maximal clique in G, and since the cover of $K(G)$ covers all of the vertices, the resulting cover by these maximal cliques must cover all edges of G.

Consider any edge clique cover of G. Each clique in this cover, forms a clique in $K(G)$ when we consider the edges of the clique in G. Since the edge clique cover of G covers each edge, it can be transformed into a clique cover of all vertices of $K(G)$. ☐

As described in the next section, the preceding result was used by Raychaudhuri [46], to find an efficient algorithms to find minimum intersection representations of the chordal graphs.

2.3.2 Chordal Graphs and Related Classes

The following theorem shown by Raychaudhuri [46] shows that for an intersection graph G, $K(G)$ is also an intersection graph of the same family, provided two conditions are satisfied. This theorem, with proof, is presented here.

Theorem 2.9 (Raychaudhuri [46]). *If G is an intersection graph of sets belonging to a family $\mathcal{F}$ of sets that is closed under intersection, such that there is an intersection representation of G as sets of $\mathcal{F}$ with the Helly property, then $K(G)$ is also an intersection graph of sets belonging to $\mathcal{F}$.*

Proof. Let $G = (V, E)$ be a graph, $\mathcal{F}$ a family of sets closed under intersection, and let $\mathcal{C} = \{S_v \in \mathcal{F} : v \in V\}$ be an intersection representation for G that satisfies the Helly property. Let $K(G) = (E, H)$ be the edge clique graph of G, and set $\mathcal{D} = \{S_u \cap S_v \in \mathcal{F} : (u, v) \in E\}$. We claim that $\mathcal{D}$ is an intersection representation of $K(G)$.

Consider any two vertices $(u, v), (z, w)$ of $K(G)$, where not all of these vertices of G need be distinct. These vertices are adjacent if and only if the vertices u, v, z, and w are contained in a common clique of G, which is true if and only if $(S_u \cap S_v) \cap (S_z \cap S_w) \neq \emptyset$ since $\mathcal{C}$ has the Helly property. Thus, $\mathcal{D}$ is an intersection representation of $K(G)$ as sets of $\mathcal{F}$. ☐

Using this theorem, we can show that many classes of graphs are closed under the edge clique graph operator simply by identifying those classes of graphs that have Helly intersection representations as sets in some family that is closed under intersection. The primary class of graphs we shall examine in this manner are the chordal graphs, which, by a result of Gavril [22], are exactly the intersection graphs of subtrees of a tree. To use Theorem 2.9, we first need the following well-known proposition that subtrees of a tree have the Helly property. This result can be found, for example, in [26].

Proposition 2.10. *If $\mathcal{T}$ is a family of subtrees of a tree, then any collection of sets of $\mathcal{T}$ has the Helly property.*

Proof. Assume on the contrary that $\mathcal{C} = \{T_1, T_2, \ldots, T_n\}$ is a minimal collection of subtrees of a tree T that does not have the Helly property. By minimality, $\mathcal{C} \setminus T_n$ has the Helly property, and notice that we must have $n \geq 3$, as a collection of two or fewer sets always has the Helly property. Since $\mathcal{C}$ is minimal, the elements of $\mathcal{C}$ must be pairwise intersecting, as otherwise, there would be elements of $\mathcal{C}$ that we could remove, and still obtain a collection without the Helly property. Consider the nonempty set given by

$$S = \bigcap_{i=i}^{n-1} T_i.$$

As this subcollection of $\mathcal{C}$ has the Helly property it must be the case that $T_n \cap S = \emptyset$, and also, no $T_i \subseteq S$, as this would imply that T_n is not disjoint from S. S forms a subtree of T since the intersection of any two subtrees remains connected.

Let u, v be any two vertices at distance one from S. If no such vertices exist, then any subtree that intersects S must either be contained in S, or must contain a common vertex w, as there must be at least one vertex outside of S. Since we have argued that no subtree is contained in S, the first case cannot occur, and the second implies that all subtrees except T_n contain w, and so we would have $w \in S$, a contradiction. Thus, we can take u, v to be distinct vertices at distance one from S.

Since u and v are at distance one from S, which is a subtree of T, there is a path from u to v though S. Also, since T_n is disjoint from S, we can either find a path from u to v through T_n, or there is some other vertex, w, at distance one from S that is on the path from u to T_n and the path from v to T_n. In either case, we find two paths from either u to v or from w to v, and this forces a cycle in T, which is a contradiction. Thus any collection of subtrees of a tree must have the Helly property. □

Then, using this proposition, we can apply Theorem 2.9, as is done in [46], to show that the edge clique graphs of chordal graphs remain chordal. This is given as the following theorem. This result is also shown using different techniques in [1] and in [45].

Theorem 2.11 ([1, 45, 46]). *If G is chordal, then $K(G)$ is chordal.*

Proof. By a result of Gavril [22], the chordal graphs are exactly the intersection graphs of subtrees of a tree. By Proposition 2.10 the subtrees of a tree have the Helly property, and so by Theorem 2.9, the edge clique graph of a chordal graph is also the intersection graph of subtrees of a tree, as the intersection of two subtrees remains a subtree. □

Finally, using this theorem and Proposition 2.8, we can apply a vertex clique covering algorithm on chordal graphs to obtain a minimum intersection representation of G. This is the approach taken by Raychaudhuri in [45], where an algorithm of Gavril [25] is applied to solve the problem. This process can be applied, for any family of sets $\mathcal{F}$ that is closed under intersection, to the class of graphs that have Helly intersection representations as sets in $\mathcal{F}$, as long as the class of graphs admits a polynomial time clique covering algorithm. An example of such a class is the Helly circular-arc graphs that do not have a pair of arcs that cover the circle. These graphs, by definition, have Helly representations as arcs. With the added restriction that no two arcs cover the circle, the intersection of any two arcs in a representation remains an arc. This added restriction does not affect the construction used by Raychaudhuri in the proof of Theorem 2.9, and so we may conclude by this theorem that for any G that is a Helly circular-arc graph with no two arcs that cover the circle, the graph $K(G)$ is also a circular-arc graph. Finally, we can apply a clique covering algorithm for circular-arc graphs [29] to find a minimum intersection representation for the graph G.

2.3.3 Comparability Graphs and Related Classes

In order to find a minimum intersection representation for a general perfect graph, we could attempt to use a similar technique to the chordal graphs. If the edge clique graphs of perfect graphs were perfect, we could use the ellipsoid algorithm for semidefinite programming to find a minimum vertex clique cover for the edge clique graph of a perfect graph, as is done by Grötschel, Lovász, and Schrijver in [28]. This approach has one fatal flaw: unlike chordal graphs, the edge clique graphs of perfect graphs need not be perfect. In fact, Figure 2.3 gives a minimum example of a cograph whose edge clique graph is not perfect. The edge clique graph of this graph, with the odd chordless cycle that demonstrates that it is imperfect, is also given. Since cographs are contained in the comparability graphs, this example rules out the direct application of this approach for that class of graphs as well.

A similar approach can be used to find an algorithm for the intersection number of comparability graphs that do not contain W_4, the wheel on four vertices, as an induced

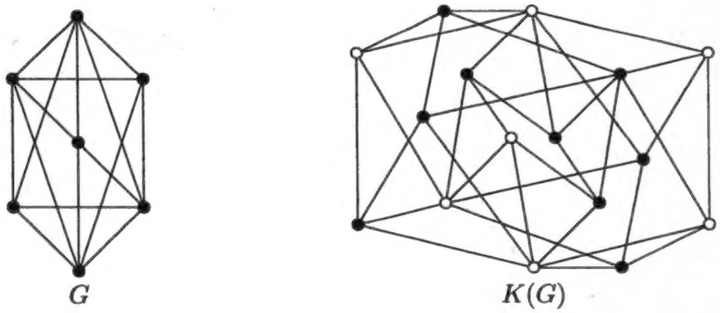

Figure 2.3: A minimum cograph G with edge clique graph that is not perfect. The white vertices of $K(G)$ form an odd chordless cycle.

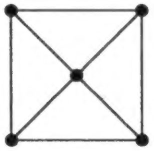

Figure 2.4: The graph W_4, the wheel on four vertices.

subgraph. The graph W_4 is given as Figure 2.4. It is shown in [46] that the edge clique graph of any W_4-free comparability graph is *weakly θ-perfect*, which is the condition that the vertex clique cover number, $\theta(G)$, is equal to the independence number, $\alpha(G)$. This condition is not directly used in the construction of the algorithm for finding the intersection number of the W_4-free comparability graphs. Instead the W_4-free condition is used to argue the correctness of a minimum-flow based algorithm that finds a minimum edge clique cover of the input graph.

2.3.4 Octahedral Graphs

In this section we investigate the size of a minimum intersection representation for the complement of a perfect matching on $2n$ vertices. This graph is interesting since it contains an exponential number of maximal cliques, which as we will see later, requires that any intersection representation that has the Helly property must have size exponential in the number of vertices in the graph. Following [43], we denote this graph O_n. An example of these graphs is given in Figure 2.5. Due to the structure of this graph, we will see that an intersection representation of O_n is closely related to an overlap representation of a clique. To see this connection, we extend the notion of overlapping, from Definition 1.2, to binary

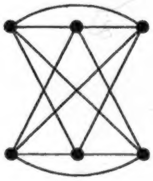

Figure 2.5: The graph O_3, the octahedral graph on three vertices.

sequences.

Given a binary sequence, $\{a_i\}_{i=1}^m$, over $\{0,1\}$, we can associate it with the index set $\{i : a_i = 1\} \subseteq \{1,2,\dots,m\}$. Using this association, we define the following set notions on binary sequences.

Definition 2.12. A pair of binary sequences *overlap* if the sets associated with them overlap. If the complements of the sequences, $\{\neg a_i\}_{i=1}^m$ and $\{\neg b_i\}_{i=1}^m$, overlap, then the two sequences are said to have overlapping complements. If two sequences overlap, and have overlapping complements, we say that they are *dually overlapping*. We will also consider subsets of $\{1,2,\dots,m\}$ to be dually overlapping if they overlap, and their complements, with respect to $\{1,2,\dots,m\}$, also overlap.

Having introduced the notion of overlapping sequences, we can show that an intersection representation for O_n immediately maps to a collection of binary sequences with an overlapping property. This will allow us to use the results that will follow in Section 3.2.1 to solve the problem of finding an intersection representation for O_n. The following lemma shows that an edge clique-cover of O_n is equivalent to a collection of dually overlapping binary sequences.

Lemma 2.13. *For any n, $\theta_e(O_n)$ is the smallest m such that there exists a collection of n binary sequences of length m that are pairwise dually overlapping.*

Proof. Given an edge clique covering of O_n, we can arbitrarily extend each clique in the cover to a maximal clique, forming a collection $\mathcal{C} = \{C_1, C_2, \dots, C_m\}$. For each nonadjacent pair of vertices in O_n, each C_i will contain exactly one vertex of the pair, by the maximality of the C_i. If we apply an arbitrary ordering on these pairs, we obtain a binary sequence of length m, for each pair of vertices, given by $\{b_i\}_{i=1}^m$, where b_i is zero if and only if the smaller vertex of the pair is in clique i. Since $\mathcal{C}$ forms an edge clique cover, each vertex in the graph will be in some clique with each other vertex in the graph, with the exception of the single vertex it is nonadjacent to. This fact ensures that for any pair of binary sequences there is some index where both sequences are zero, and another where both sequences are one. In addition, if one sequence contains another, in the sense that the set of indices of nonzero elements of one sequence contains such a set constructed from the other sequence, then the smaller vertex of one pair cannot be in the same clique as the larger vertex of the other, contradicting the assumption that C forms an edge clique cover. Thus, from an edge clique covering we can construct a set of $\theta_e(O_n)$ binary sequences that pairwise dually overlap.

The same construction also functions in reverse. Given a set of n binary sequences that are dually overlapping, $\mathcal{B} = \{\{b_{ji}\}_{i=1}^{m} : 1 \leq j \leq n\}$, we can construct an edge clique cover for O_n. To do this, where the vertices of O_n are $\{0, \ldots, 2n-1\}$, and the nonadjacent pairs are given by $(k, k+1)$, for even k, we form the ith clique by

$$C_i = \{2j + b_{ji} : 0 \leq j \leq n-1\},$$

for all $1 \leq i \leq m$. Notice that, once again, since any two of the sequences overlap, and so do their complements, each vertex will be adjacent to both halves of each other nonadjacent pair, forming an edge clique cover for O_n with size given by the length of the binary sequences.

Thus, since an edge clique cover can be transformed into a set of binary sequences with the desired property, of the same size, the length of the binary sequences we need to form a collection of n such sequences is exactly $\theta_e(O_n)$. □

We can reduce the problem of finding a collections of dually overlapping binary sequences to yet another problem: the problem of finding a minimum overlap representation for a clique. Phrased in terms of binary sequences, this problem seeks a maximum collection of overlapping binary sequences.

Lemma 2.14. *A maximum set of binary sequences of length $m+1$ that are dually overlapping has the same size as a maximum set of binary sequences of length m that are pairwise overlapping.*

Proof. Given a set of binary sequences that are pairwise overlapping and complement overlapping, notice that we may take the complement of any sequence without affecting the overlapping properties. We can then find a collection of the same size where the first bit of every sequence is 0, and so the remaining m bits in every sequence form a collection that is pairwise overlapping.

In the other direction, given any set of binary sequences of length m that are pairwise overlapping, we can add a 0 to the end of every sequence to obtain a set of binary sequences of length $m + 1$ that pairwise dually overlap. □

Next we simply view the binary sequences as characteristic vectors of subsets of the set $\{1, 2, \ldots, m\}$, and we are able to apply the results of Section 3.2.1 to the problem of finding the minimum intersection representation for O_n. Corollary 3.5, which is proven independently of the results in this section, regarding the size of the minimum overlap representation of K_n, and the previous lemma immediately imply the following corollary.

Corollary 2.15. *For any* $n \geq 1$,

$$\theta_e(O_n) = \min\left\{m : n \leq \binom{m}{\lfloor \frac{m+2}{2} \rfloor} + 1\right\}.$$

Proof. By Lemmas 2.13 and 2.14, any intersection representation of O_n can be used to form a collection of dually overlapping sequences, of length m, of the same size, which can in turn be used to form an overlap representation for K_{n-1}, as we can view the overlapping binary sequences of Lemma 2.14 as characteristic vectors of subsets of $\{1, 2, \ldots, m\}$. These transformations also function in reverse, and so, the minimum intersection representation for O_n has size given by the minimum overlap representation of K_{n-1}, which is, by Corollary 3.5,

$$\min\left\{m : n - 1 \leq \binom{m}{\lfloor \frac{m+2}{2} \rfloor}\right\},$$

as desired. □

In the language of Section 3.2.1, the quantity $\binom{m}{\lfloor \frac{m+2}{2} \rfloor}$ is given by the function $S(1, m)$, which is given by Definition 3.3. The equivalence of these two quantities is shown by Milner's Theorem, which is stated here as Theorem 3.4. In addition to these definitions, using the results of Section 3.2.3, we can immediately apply Theorem 3.10 to show that $\theta_e(O_n) \in \Theta(\log n)$. Furthermore, as we can construct an overlap representation for K_{n-1} in linear time, as discussed in Section 3.2.3.3, and the transformations in Lemmas 2.13 and 2.14 can be performed in linear time, we can find a minimum edge clique cover, and thus a minimum intersection representation of O_n, in linear time.

2.3.5 Extending an Intersection Representation

Given an intersection representation for a graph, it is simple to decide if we can, given a vertex to add to the graph, extend the intersection representation without increasing the size of the representation. While this may seem to be an uninteresting problem, the complexity of the problem forms a contrast to the **NP**-completeness results that we obtain in Sections 3.3.1 and 4.3.1 for the overlap and containment versions of this problem, respectively. A more formal version of the problem, which parallels the problems we shall encounter later, is given by:

Problem. The INTERSECTION EXTENSION problem is defined as:

Instance: A graph, $G = (V, E)$, an intersection representation $\mathcal{C} = \{S_v : v \in V\}$ of G, and a set $A \subseteq V$.

Question: Is there is a set $S \subseteq \bigcup_{v \in V} S_v$ that intersects S_v if and only if $v \in A$?

This problem is quite restrictive. We are unable to alter in any way the intersection representation given as part of the input. We can use this inflexibility to build a polynomial-time algorithm that decides the problem. The crucial observation is that, given any element in the representation, we can only add this element to the set S if the new vertex we are adding is adjacent to all other vertices that also have this element in the set associated with them. In terms of an edge clique cover this is simply the observation that we cannot add a vertex to a clique unless it is adjacent to all vertices in the clique. Since there is no penalty in the problem for making the set S needlessly large, we can take the greedy approach of adding every element we are able to add to the set S. This construction can be performed in polynomial time in the size of the representation and the size of the graph, as we can, for each element in the representation, decide if we can add it to the set S by scanning all sets of the representation, and the set A, which forms the neighbourhood of the new vertex. More formally, for G, $\mathcal{C}$, and A as given in the problem, we consider each element in $\bigcup_{v \in V} S_v$. For the element i, the set $N_i = \{v : i \in S_v\}$ forms a clique in S, and we can only add i to S if $N_i \subseteq A$, which is exactly the condition that the vertex we are extending the representation with also extends the clique of G given by N_i. This condition is satisfied if and only if we can add the element i to S, and it can be verified in time polynomial in the size of the representation and the size of the graph. After we have done this for each element in the representation, we simply need to examine each of the edges that adding the new vertex will create, to ensure that the sets assigned to each endpoint intersect. If there is an edge that we have not covered in this manner, the algorithm rejects, as we have added every element possible, without altering the rest of the representation, to the set S. If there are no uncovered edges, the algorithm accepts, as we have found a set S that satisfies the requirements of the problem.

2.4 Bounds on the Size of an Intersection Representation

Minimum intersection representations, in general, can have size quadratic in the size of the graph. A tight upper bound on the number of elements needed in any intersection representation is given in a 1966 paper of Erdős, Goodman, and Pósa, and the statement of the theorem appears below.

Theorem 2.16 (Erdős, Goodman, and Pósa [16]). *If G is any graph with* n *vertices, then there is an intersection representation of G with at most* $\lfloor n^2/4 \rfloor$ *elements. Furthermore,* $\lfloor n^2/4 \rfloor$ *is the smallest such number.*

The bound in this theorem is tight, as the number of cliques required to form an edge clique cover of a complete bipartite graph, as shown in [16], is given by the number of edges in the graph. For even n, if each partition has $n/2$ vertices, any representation must have exactly $n^2/4$ elements. If n is odd, if one partition has $(n-1)/2$ vertices, and the other $(n+1)/2$, then any intersection representation must have exactly

$$\frac{(n+1)(n-1)}{4} = \frac{n^2-1}{4} = \left\lfloor \frac{n^2}{4} \right\rfloor$$

elements. The complete bipartite graph, as we will see in Sections 3.4 and 4.4, has an overlap number of three, and containment number that is at most linear in the number of vertices in the graph.

In the next section, we examine the size of an intersection representation that must also satisfy the Helly property. This will turn out to be directly related to the number of maximal cliques contained in a graph, which will lead to the study of those classes of graphs that have a polynomially bounded number of maximal cliques.

2.5 Representations with the Helly Property

An interesting group of graph classes that has received relatively little study are those classes that have a polynomially bounded number of maximal cliques. The study of these classes of graphs was initiated by Prisner in [43], where they are referred to as the graph classes with *few cliques*. Many of the graph classes that are commonly studied admit such a bound, and so general results about these classes of graphs can be applied in many places.

One of the essential properties of the interval graphs and the chordal graphs is that they have intersection representations that satisfy the Helly property. This property underlies many of the properties that these graphs have, and so it is natural to consider those representations for graphs that have the Helly property, as from these representations we may be able to infer properties of the graphs they represent. We have seen in Section 2.1 that any graph has a Helly intersection representation but unfortunately some of these representations may be quite large. We obtain interesting classes of graphs if we restrict our attention to those graphs that have "small" Helly intersection number.

Surprisingly, as we will see in Section 2.5.1, if we restrict our attention to those classes which have polynomial sized intersection representations that satisfy the Helly property, we obtain exactly the same graphs as those with a polynomially bounded number of maximal cliques. In addition, by using an algorithm to find all maximal cliques in a graph in time polynomial in the number of maximal cliques, such as the algorithm in [31] or [61], we can obtain a minimum intersection representation satisfying the Helly property in time polynomial in the size of the representation, which may be exponential in the number of vertices of the graph. Solving problems on graphs with few cliques is then polynomially equivalent to solving the same problems on graphs where we are given as input a Helly intersection representation, provided this representation is not exponentially large in the size of the input graph. We can omit the case where the Helly intersection number is of size exponential in the number of vertices in the input graph, as the complexity questions we consider are uninteresting in this case. This is because given as input to an algorithm a graph and an exponentially large Helly intersection representation, we can run an algorithm that takes time exponential in the size of the graph, while still using time that is polynomial in the size of the input. Since the problems we consider are in **NP**, and a nondeterministic algorithm can be simulated by a deterministic one in exponential time, the problems we will consider will all be "efficiently" solvable if we are given an exponential sized Helly intersection representation. In addition to this, the hardness results we will show will be for the class of all graphs containing no more than some bounded number of maximal cliques. If we increase this bound to allow graphs with exponentially many maximal cliques the same results will still apply, as the hard instances from the smaller class are still contained in the larger class. Thus, the complexity of the problems we consider is effectively independent of those graphs with exponential Helly intersection number, and so we consider only the case where the number of maximal cliques is bounded by some polynomial in the size of the graph.

As we will see in Corollary 2.22, the size of a minimum Helly intersection representation for a graph is exactly equal to the number of maximal cliques in the graph. We will be primarily concerned, however, with graphs that have a polynomially bounded number of maximal cliques, as we feel that the results we present are more naturally phrased in this

Problem	**Complexity**	**Bound**
CLIQUE	$O(nmp(n))$	
COLOURABILITY	**NP**-complete	$4n^2$
DOMINATING SET	**NP**-complete	$\frac{7}{3}n - 6$
EDGE CLIQUE COVER	**NP**-complete	$(m + 1)n^3$
EDGE CLIQUE k-COVER	$O(n^2p(n)^k)$	
HAMILTONIAN CYCLE	**NP**-complete	$\frac{7}{3}n - 6$
HAMILTONIAN PATH	**NP**-complete	$\frac{7}{3}n - 6$
INDEPENDENT SET	**NP**-complete	$4n^2$
k-CLIQUE PARTITION	**NP**-complete	n^k
k-COLOURABILITY	**NP**-complete	$\frac{7}{3}n - 6$
VERTEX CLIQUE COVER	**NP**-complete	n^3
VERTEX CLIQUE k-COVER	$O(n^2p(n)^k)$	
VERTEX CLIQUE k-PARTITION	$O(n^2p(n)^k)$	
VERTEX CLIQUE PARTITION	**NP**-complete	n^3
VERTEX COVER	**NP**-complete	$4n^2$
WEIGHTED CLIQUE	$O(nmp(n))$	

Table 2.1: Complexity results on graphs with n vertices, m edges, and no more than $p(n)$ maximal cliques. The **NP**-completeness results hold for the class of graphs that do not have more maximal cliques than the given bound.

manner. These results will include both polynomial time algorithms and reductions from **NP**-complete problems for the class of graphs with no more than some given number of maximal cliques.

The problems that are solvable in polynomial time on graphs with a polynomial number of maximal cliques are ones that might be expected: the problems CLIQUE, WEIGHTED CLIQUE, and VERTEX CLIQUE k-COVER. In contrast, many problems remain hard on this class, including INDEPENDENT SET, COLOURABILITY , even if the number of colours is restricted to a constant, and, somewhat surprisingly, EDGE CLIQUE COVER and VERTEX CLIQUE COVER. This last fact implies that it remains hard to find a minimum intersection representation, even when given a minimum Helly intersection representation. Table 2.1 lists the complexity results on these problems considered here, for graphs with few cliques.

Many of these problems, such as CLIQUE, COLOURABILITY, and INDEPENDENT SET are well known, and are not defined here. The definitions of these problems can be found in [21] or in Appendix A.

Some of these results are due to the wide range of graph classes which turn out to have a polynomial bound on the number of maximal cliques, and thus a polynomially bounded Helly intersection representation. These classes include chordal graphs, interval graphs, graphs of boxicity k, and planar graphs. From this range of classes, it is apparent that many problems are **NP**-complete on graphs with a polynomial number of maximal cliques, simply due to known **NP**-completeness results for graph classes which have this property.

Previous work on graphs with a polynomial number of cliques is sparse. Prisner [43] gives bounds on the number of maximal cliques in classes of graphs, and provides a characterization for any hereditary class of graphs with a polynomially bounded number of maximal cliques. Balas and Yu [2] prove a similar bound, but they do not consider characterizing those graphs which have a polynomial number of maximal cliques, as their focus is on solving a different problem. Predating these works is a 1965 paper by Moon and Moser [40], which gives an upper bound of $3^{\lceil n/3 \rceil}$ maximal cliques for any n vertex graph, and provides a tight example to show that this bound is achieved.

In order to make a reasonable definition of the graphs with a polynomial number of maximal cliques, we follow [43] in defining this concept only on classes of graphs.

Definition 2.17. A class $\mathcal{G}$ of graphs has *few cliques* if there is a polynomial $p(n)$ such that for any $G \in \mathcal{G}$ with n vertices, there are no more than $p(n)$ maximal cliques in G.

Notice that, by this definition, any finite class of graphs has few cliques. We will also use the notion that a class of graphs has polynomially bounded Helly intersection representation in order to refer to a class of graphs with few cliques, as we will show in Section 2.5.1 that these notions are equivalent. As examples of graphs in these classes, K_n, the clique on n vertices, has only a single maximal clique, and n maximal independent sets, each consisting of a single vertex. Thus, the class of complete graphs has few cliques (and also few independent sets). Similarly, any clique in a triangle free graph is simply an edge of the graph. Since trees and bipartite graphs are triangle free, we see that there are exactly $n-1$ maximal cliques in any tree on n vertices, and there are no more than $\binom{n}{2}$ maximal cliques in a bipartite graph. These bounds demonstrate that these classes of graphs have few cliques.

In the next section we will see more classes of graphs which have few cliques, as well as some characterizations of those classes of graphs that have few cliques. The following section, Section 2.5.2, uses these characterizations to find some classes of graphs which have such a bound on the number of maximal cliques. Section 2.5.3 presents some complexity results which include both polynomial time algorithms and **NP**-completeness results for graphs with a polynomially bounded number of maximal cliques.

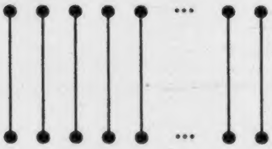

Figure 2.6: The complement of this graph on n vertices contains $2^{n/2}$ maximal cliques.

2.5.1 Characterizations of Graphs with Few Cliques

An early bound on the number of maximal cliques in a graph was demonstrated in a 1965 paper by Moon and Moser [40], where they show that a graph cannot have more than $3^{\lceil n/3\rceil}$ maximal cliques. In addition, Moon and Moser also show that this bound is tight, by providing a graph with $3^{n/3}$ cliques, which is simply the complement of the graph consisting of $n/3$ disjoint triangles. This bound leaves open the problem of determining which classes of graphs do admit a polynomial bound on the number of maximal cliques in any graph in the class. When examining the structure of graphs with a large number of maximal cliques, it seems to be the case that induced subgraphs of a particular form are a particular problem. These are the graphs of the form $O_t = \overline{tK_2}$ for some $t \in \mathbb{N}$. This graph can be obtained by taking the complement of a perfect matching on $2t$ vertices. To find a maximal clique in such a graph, we select one of each pair of matched vertices. Since there are 2^t ways to choose one vertex from each matched pair and since every maximal clique contains exactly one vertex from each matched pair, there are exactly 2^t maximal cliques in the graph O_t.

In order to extend an algorithm for the maximum weighted clique problem, Balas and Yu [2] prove the following bound on the number of maximal cliques a graph can have, based on the largest induced subgraph of the form O_t.

Theorem 2.18 (Balas and Yu [2]). *For any connected graph $G = (V, E)$, where δ is the number of pairs $(u, v) \in V \times V$ with $d(u, v) = 2$, and t is the largest integer such that G contains an induced subgraph isomorphic to O_t, the number of cliques in the graph is bounded by*

$$2^t \leq |\mathcal{K}(G)| \leq \delta^t + 1.$$

Notice that there can be no more than n^2 pairs of vertices at distance two, and so, in the case that t is bounded by some constant for a class of graphs this theorem demonstrates that there is a polynomial bound on the number of maximal cliques in graphs in the class. One immediate example of this is the class of chordal graphs. This theorem can be applied, as O_2 is exactly a chordless cycle of length four. In this case we can have δ as large as

$(n-1)^2$, by noticing that $K_{1,n-1}$, a star on n vertices, contains no cycles, so it is a chordal graph, and that any two vertices neither of which is the center of the star are at distance two. A simple counting argument shows that this also provides an upper bound in the number of pairs at distance two. Then, applying the theorem we obtain, for any chordal graph

$$2 \leq |\mathcal{K}(G)| \leq (n-1)^2 + 1 < n^2,$$

and so we know that the chordal graphs have few cliques. In fact, this bound is far from tight, as the following proposition, first noted in 1965 by Fulkerson and Gross [17], demonstrates.

Proposition 2.19 ([17]). *If G is a chordal graph with n vertices, then $|\mathcal{K}(G)| \leq n$.*

Another partial characterization of the graph classes with few cliques, similar to Theorem 2.18 is shown by Prisner in [43], where he considers classes of graphs where O_t is a forbidden induced subgraph, deriving upper bounds on the number of cliques in such graphs. A corollary to these bounds, which is simpler to state, characterizes the hereditary classes of graphs which have few cliques.

Theorem 2.20 (Prisner [43]). *A hereditary class $\mathcal{C}$ has few cliques if and only if for some constant t, $O_t \notin \mathcal{C}$.*

As Prisner observes in [43], this theorem implies that the K_k free graphs have few cliques. This class is clearly hereditary, as removing a vertex cannot add an induced K_k to the graph. In addition, the graph O_t contains an induced K_t (in fact, it contains 2^t of them), so the graph $O_k = \overline{kK_2}$ is not K_k free. Hence, we can conclude by the theorem that this class of graphs has few cliques. The same result can be obtained by a counting argument.

In addition to these results relating the number of maximal cliques in a graph to the subgraphs of the form O_t, we can turn, as promised in the previous section, to the intersection representation of a graph in order to obtain a characterization of the classes of graphs which have a polynomially bounded number of maximal cliques. To do so, we consider the minimum size of any intersection representation which satisfies the Helly property. In order to relate the number of maximal cliques in a graph and the size of a Helly intersection representation, we turn to a theorem on hypergraphs. The hypergraph definitions used here can be found in [3] and [5]. A *hypergraph* $H = (V, \mathcal{E})$ is a generalization of a graph, where the elements of the edge set $\mathcal{E}$, are now permitted to be arbitrary subsets of the vertices of the graph, in contrast to the case for graphs, in which the edges are subsets of size two.

The *dual* of a hypergraph $H = (V, \mathcal{E})$ is given by $H^* = (\mathcal{E}, \{E_v : v \in V\})$, where E_v is the set of hyperedges in $\mathcal{E}$ that contain the vertex v. The *2-section*, $(H)_2$ of a hypergraph $H = (V, \mathcal{E})$ is the graph $G = (V, E)$, where $(u, v) \in E$ if and only if there is an edge $e \in \mathcal{E}$ such that $u, v \in e$. We will say that a hypergraph $H = (V, \mathcal{E})$ is Helly when the edge set $\mathcal{E}$ satisfies the Helly property. Another property we will need is that of conformality; a hypergraph H is *conformal* if each clique of $(H)_2$ is contained in an edge of H. Having introduced this notation, we can introduce a theorem which can be found in [3], although it is there attributed to Gilmore without a reference.

Theorem 2.21 (see Berge [3, page 396]). *A hypergraph is Helly if and only if its dual is conformal.*

At first glance this theorem might not seem to relate to the notion of a minimum Helly intersection representation of a graph, but since the 2-section of the dual of a hypergraph H is simply the graph formed by the intersection of the hyperedges of H, and we can see the hyperedges of H as sets in an intersection representation, Theorem 2.21 shows that all cliques of the intersection graph must be contained in sets of any Helly intersection representation of that graph. Furthermore, by the theorem, we know that the set of all maximal cliques of the graph will suffice, as this will yield a conformal hypergraph, when the edges of the hypergraph are simply the maximal cliques of the graph. This is summarized in the following corollary to Theorem 2.21.

Corollary 2.22. *For any graph G, the number of maximal cliques in G is equal to the Helly intersection number of G.*

Using this corollary, we see that a class of graphs has at most $p(n)$ cliques if and only if there is an intersection representation that satisfies the Helly property using at most $p(n)$ elements, for any member of the class. This can be used in both directions. We can use this to show that classes of graphs have few cliques, or we can use this to find an intersection representation of a class of graphs. For example, this gives an intersection representation that satisfies the Helly property for K_k free graphs. In addition to this, we can generate this representation in polynomial time. For any graph $G = (V, E)$, with $n = |V|$ and $m = |E|$, we can find all maximal cliques in $O(nm\,|\mathcal{K}(G)|)$ time, using an algorithm from [31] or [61]. We can then build the set of cliques a vertex is contained in within time polynomial in the number of cliques and vertices in the graph. Since there is some polynomial $p(n)$ which bounds $|\mathcal{K}(G)|$, this algorithm runs in time polynomial in the size of the graph.

Using these results, we can identify several classes of graphs which have few cliques. These classes will be used later when discussing the complexity of a few problems on classes of graphs with few cliques. The next section presents some of these classes.

2.5.2 Graph Classes with Few Cliques

Many classes of graphs admit a polynomial bound on the number of maximal cliques that any member of the class may contain. In the next section we will see some complexity results for any class of graphs with few cliques, but first, in this section, we list many of the common classes of graphs with few cliques.

As we have already seen, it is clear that the complete graphs K_n for any n have a single maximal clique, which is the one containing all the vertices of the graph. Another class of graphs which has few cliques are the K_k free graphs. Notice that in such a graph there cannot be any cliques of size larger than k, as these cliques would have K_k as an induced subgraph. Then, a crude bound on the number of cliques (maximal or not) is obtained by taking the number of ways to choose fewer than k vertices from the n vertices in the graph. This is given by

$$\sum_{i=1}^{k-1} \binom{n}{i} \in O(n^{k-1}),$$

and so there is a polynomial bound on the number of cliques in the K_k free graphs. A more careful analysis done in [43] yields a bound of $\max\{n, n\Delta^{k-2}/2^{k-2}\}$, where Δ is the maximum degree of any vertex in the graph, which is still $O(n^{k-1})$ in the case that $\Delta \in O(n)$.

This result can be used to bound the number of cliques in several classes of graphs. Trees and bipartite graphs are triangle free, and so there are at most a quadratic number of maximal cliques in any graph in these classes. More precisely, the cliques in a triangle free graph are exactly the edges of the graph, and so there are $n-1$ maximal cliques in any tree, and no more than $\binom{n}{2}$ maximal cliques in any bipartite graph. One more class of graphs we can apply this to are the planar graphs. Since these graphs are K_5 free (this is a trivial consequence of Kuratowski's Theorem [36]), we know immediately that there are at most $O(n^4)$ maximal cliques in a planar graph, by the preceding argument. Once again, a more careful analysis produces a better bound, and in this case a much better bound. The results of this analysis is given as the following theorem.

Theorem 2.23 (Prisner [43]). *If G is a planar graph with n vertices, then $|\mathcal{K}(G)| \leq \frac{7}{3}n-6$.*

We have already seen the case of chordal graphs, but armed with Corollary 2.22, we are provided with another method of showing that this class has few cliques. Once again, by a result of Gavril [22], the chordal graphs are exactly the intersection graphs of subtrees in a tree. Also, as we have seen in Proposition 2.10, subtrees of a tree have the Helly property, which implies, by Corollary 2.22, that the chordal graphs have a polynomial bound on the number of maximal cliques they can contain, since the subtree representation can be constructed using a polynomial sized tree. The interval graphs are a subclass of the chordal graphs, and so we know that this class of graphs also has few cliques.

We can extend this idea to a generalization of the interval graphs. The graphs of boxicity k, introduced in [48], are those graphs that can be represented as the intersection graphs of axis-aligned 'boxes' in k dimensional space. The case of primary interest here is the class of boxicity-2 graphs, which can be represented as the intersection graphs of axis-aligned rectangles in the plane, since an **NP**-completeness result on this class will be used in Section 2.5.3. It is noted in [52] that there are at most $O(n^k)$ maximal cliques in a graph of boxicity k, as boxes in $\mathbb{R}^k$ have the Helly property. We can be more specific by noticing that to form an intersection representation for a graph with boxicity k, we need only include in our representation the points in $\mathbb{R}^k$ that are vertices of boxes. This uses $(2n)^k$ points, and so there are at most this many maximal cliques in any graph with boxicity k, by Corollary 2.22.

Using the fact that these classes of graphs have a polynomially bounded number of maximal cliques, we can immediately conclude that several problems are hard on graphs with few cliques. This is done in the next section.

2.5.3 Complexity Results on Graphs with Few Cliques

Before introducing some specific complexity results we make some general remarks, in Section 2.5.3.1, on the form that such results must take. We consider recognizing a graph that satisfies a given bound on the number of maximal cliques, and we also specify exactly the class of graphs that we can find **NP**-completeness results for. After introducing the notions we will use, we present a number of complexity results in Section 2.5.3.2.

Some interesting problems on classes of graphs with a polynomially bounded number of cliques are the clique partitioning and covering problems. When considering these problems on graphs with few cliques, we can examine whether the hardness of the problem lies in finding cliques in an exponentially large collection, or in choosing the cliques to build a minimum partition from a polynomially sized set. As we will see in Sections 2.5.3.3, 2.5.3.4, and 2.5.3.5, the majority of these problems remain hard with a restricted number of maximal cliques.

2.5.3.1 Recognition, Robustness, and NP-completeness

The problem of recognizing a graph that satisfies a particular bound is simply, given some polynomial bound $p(n)$, does the input graph have no more than $p(n)$ maximal cliques? We can equivalently ask, by Corollary 2.22, if a graph has a minimum Helly intersection representation of size no more than $p(n)$. Without a bound, counting the number of maximal cliques in a graph is **#P**-complete, which is shown implicitly for general graphs in [63], and explicitly for some restricted classes of graphs in [62]. These results indicate that the decision version of the problem is likely not contained in **NP**, since by Toda's Theorem [57], as mentioned in Section 1.2.2, this inclusion would collapse the polynomial hierarchy.

Fortunately, while this is a hard counting problem, if we are also given a polynomial bound $p(n)$, it is in fact solvable in polynomial time, by using an algorithm that enumerates all maximal cliques in a graph. If the algorithm we use has what is referred to in [31] as the *polynomial delay* property, then we will be able to simply run the algorithm to enumerate the cliques, and stop if we find more than $p(n)$ cliques. The polynomial delay property that allows us to do this is simply the property that the length of time before the first clique is generated and the length of time between the algorithm generating any two successive cliques are each bounded by a polynomial in the size of the graph. Since polynomials are closed under multiplication, within a polynomial amount of time we will then either have generated too many cliques, whence we would stop the computation and reject, or we will have enumerated all of the cliques in the input graph, at which time the algorithm accepts. Fortunately, the algorithm found in [61] has this property, as does the much simpler algorithm found in [31], which can be asymptotically slower than the more complicated algorithm.

We can use this recognition algorithm to ensure that any polynomial time algorithm for graphs with few cliques is *robust*, in the sense introduced in [52], where a robust algorithm is one that when given input not in the class of graphs the algorithm is designed for, the algorithm will either notice this and reject, or continue computing the correct output, even though the graph is not of the expected class. This addition of robustness, however, only works if we are provided with a specific polynomial bound, $p(n)$, on the number of maximal cliques in any member of class of graphs the algorithm is to operate on. Since all the algorithms we present begin with generating all maximal cliques of the input graph, we can simply modify this step, by generating at most $p(n)$ cliques, and indicating that the input was not in the class of graphs if we generate more than $p(n)$ cliques. This satisfies the requirement of a robust algorithm, and so, given in advance a polynomial bound, which is, given a class of graphs with few cliques to operate on, and a bound on the number of cliques in any graph in the class, we can modify any of the algorithms that follow to produce robust algorithms.

In addition to these considerations, we must be precise about exactly which classes of graphs that we provide **NP**-completeness results for. We consider the class of all those graphs containing not more than $p(n)$ maximal cliques, for a given polynomial $p(n)$. It is important to note that we cannot provide **NP**-completeness results for every class of graphs with no more than $p(n)$ cliques, as, for instance, the class of graphs containing only the single graph K_1 has few cliques, but since it contains only a single graph of constant size, there exist trivial algorithms that can be used solve any decision problem on this class of graphs in constant time. Since the graph classes we have shown to have few cliques are contained in the class of graphs with not more than $p(n)$ cliques, for some choice of $p(n)$, **NP**-completeness results for these classes will directly imply the **NP**-completeness of the same problem on the class of graphs with not more than $p(n)$ cliques. This is a strategy employed to show the hardness of several problems in the next section.

2.5.3.2 Simple Complexity Results

Having argued that a number of graph classes have few cliques, we can now present a number of **NP**-completeness results on graphs with few cliques. However, before discussing **NP**-complete problems, there are two problems which are rendered trivial by the existence of a polynomial bound on the number of maximal cliques in an input graph. These problems are CLIQUE, the problem of finding the largest clique, and WEIGHTED CLIQUE, the problem of finding the clique with the largest weight in a vertex-weighted graph. In order to solve these problems, on a graph G, given that there is a polynomial bound $p(n)$ on the number of maximal cliques in G, we can simply apply the algorithm in [61] to find all maximal cliques in time $O(nmp(n))$, since $|\mathcal{K}(G)| \leq p(n)$. The algorithm does not need to have access to this bound, but it cannot be made robust if the bound $p(n)$ is not known. Then, to solve the maximum clique problem we can simply iterate over the list of all maximal cliques, selecting the largest one. Calculating the size of a clique in the list can easily be done in $O(n)$ time. Thus, the total time required to scan the list is bounded by $O(np(n))$ as there are at most $p(n)$ cliques in the list. This has total cost bounded by the maximal clique finding algorithm, which is $O(nmp(n))$. A very similar algorithm solves the vertex-weighted maximum clique problem. We again build a list of all maximal cliques. Then, for each clique in the list we examine the vertices in the clique, discarding any that have negative weights, and for the resulting list of cliques we choose the one with largest weight. If the process of removing vertices leaves us with only empty graphs, we instead search for the vertex with maximum weight in the graph and use this single vertex as the output clique. This removal process can be completed in $O(np(n))$ time, by considering each vertex of every clique, and finding the maximum can also be completed in $O(np(n))$ time. Thus, the total time for this algorithm is also bounded by the time to find the maximal cliques of G, which can be done in $O(nmp(n))$ time. This algorithm will correctly find the maximum weighted clique, as the maximum weighted clique must be contained in some maximal clique of the graph. This is because any vertex with nonnegative weight will not decrease the weight of a clique, and so by removing vertices with negative weights, we will not overlook a maximum weighted clique, as such a clique cannot contain any vertices with negative weights, as long as G has a vertex of positive weight. If all vertices of G have negative weights, we can choose the vertex with largest weight as the maximum weighted clique. These results are summarized in the following proposition.

Proposition 2.24. *If $\mathcal{G}$ is a class of graphs with few cliques, and $G \in \mathcal{G}$ has n vertices, m edges, and weights associated with the vertices, then there is an algorithm to find the maximum weighted clique of G in $O(nm\,|\mathcal{K}(G)|)$ time.*

In (widely believed) contrast to these problems that are solvable in polynomial time on classes of graphs with few cliques, there are problems which are **NP**-complete on these classes of graphs. Many of these **NP**-completeness results can be trivially seen by providing a class of graphs with few cliques for which it is known that the problem is **NP**-complete. Such a class is the class of boxicity-2 graphs, which have no more than $4n^2$ cliques, as observed in [52]. For this class of graphs it is known that the problem COLOURABILITY is **NP**-complete [37] and the problem INDEPENDENT SET is **NP**-complete [47]. These results imply the **NP**-hardness of these two problems for any class of graphs that includes the boxicity-2 graphs, such as those graphs with no more than $4n^2$ maximal cliques. These problems are also **NP**-complete for this class, as they are in **NP** for general graphs, so the same algorithms can be used to decide them in the case of any restricted class of graphs.

Once we know that the maximum independent set problem is **NP**-complete for a class of graphs, then immediately we know that VERTEX COVER is also complete for the same class of graphs, as it is well-known (see [64], for example) that the size of a minimum vertex cover is given by $n - \alpha(G)$.

There are three other problems we can obtain the **NP**-completeness of without significant effort. These problems are HAMILTONIAN CYCLE, HAMILTONIAN PATH, and DOMINATING SET. The **NP**-completeness of the Hamiltonian cycle and Hamiltonian path problems on planar graphs is shown in [20], and the dominating set problem on planar graphs is shown **NP**-complete in [21]. Thus, by the bound in [43] which limits the number of maximal cliques in a planar graph to $\frac{7}{3}n - 6$ we know that these problems are **NP**-complete for the class of graphs with no more than $\frac{7}{3}n - 6$ maximal cliques, where once again, we have containment in **NP** by the **NP**-completeness of these problems for general graphs.

2.5.3.3 k-Colourability

In Section 2.5.3.2, we saw that having a polynomial bound on the number of maximal cliques does not help the problem of finding the minimum colouring of a graph; in this section, we consider the problem of colouring a graph with no more than some constant number of colours, k, where $k \geq 3$. This is formalized as the well-known k-COLOURABILITY problem, which is defined in Appendix A.

3-COLOURABILITY is known to be **NP**-complete on planar graphs [19], and thus, by the bound given in [43], the problem is also **NP**-complete for the class of graphs with no more than $\frac{7}{3}n - 6$ maximal cliques, where n is the number of vertices in the graph. This result does not extend, on planar graphs, to values of k larger than 3, as it is known that any planar graph can be coloured using only four colours. However, if we are willing to lose planarity we can, without sacrificing the bound on the number of maximal cliques, use a simple reduction to reach larger values of k. We will reduce k-colourability to $k + 1$-colourability, and thus, by induction on k, we can show the following theorem.

Theorem 2.25. k-COLOURABILITY, *for* $k \geq 3$, *is* **NP**-*complete on the class of graphs with no more than* $\frac{7}{3}n - 6$ *maximal cliques.*

Proof. We can use a simple reduction from the k-colouring problem to the $k + 1$-colouring problem to form an inductive proof, as the 3-colourability problem is **NP**-complete for planar graphs. We take a graph $G = (V, E)$, an instance of the k-colourability problem, and create a graph G' by adding a single universal vertex. That is, where v is a vertex not in V, we create the graph $G' = (V', E')$, where $V' = V \cup \{v\}$ and $E' = E \cup \{(u, v) : u \in V\}$.

To see that $|\mathcal{K}(G)| = |\mathcal{K}(G')|$, notice that any maximal clique in G forms a maximal clique in G' when we add the new vertex, and that any maximal clique of G' must contain the vertex v, as it is adjacent to every vertex in the graph, and since this clique is maximal, it must form a unique maximal clique in G when the vertex v is removed. In addition, the construction of the graph G' can clearly be performed polynomial time.

To see that this reduction is correct, take an instance of the k-colourability problem that is colourable with no more than k colours. We can colour the vertex added by the transformation with a new colour, and we will end up with a valid $k+1$-colouring of the graph. On the other hand, consider any $k+1$-colouring of the instance given by the transformation. The universal vertex that was added must be coloured with a unique colour, as it is adjacent to every vertex of the graph, and so if we remove this vertex, we obtain a k-colouring of the remaining graph, which is exactly the instance of the k-colourability problem that was used as input for the transformation. Therefore, by induction on k, for $k \geq 3$, this demonstrates the **NP**-hardness of the k-colourability problem on the class of graphs with not more than $\frac{7}{3}n - 6$ maximal cliques, for any constant $k \geq 3$ and we have **NP**-completeness on this classes of graphs, by the **NP** algorithm for the problem on general graphs. □

2.5.3.4 k-Clique Partition

In this section we examine the problem of partitioning the vertices of a graph into disjoint cliques of size k, for some constant k.

Problem. The k-CLIQUE PARTITION problem is defined as:

Instance: A graph, $G = (V, E)$.

Question: Can the vertices of the graph G can be partitioned into cliques of size k?

This problem can be shown to be **NP**-complete, for any $k \geq 3$, by a similar reduction to the one used in Section 2.5.3.3. The case $k = 3$ can be shown **NP**-complete by a reduction from the exact cover by 3-sets problem. This is done by Garey and Johnson in [21], where the exact cover by 3-sets problem is used to show the **NP**-completeness of 3-CLIQUE PARTITION (which they call the triangle partition problem), but most importantly for our purposes, the reduction they use produces a K_4 free graph. Thus, this result can be immediately extended to the 3-CLIQUE PARTITION problem on graphs with no more than $p(n)$ cliques, for some cubic polynomial $p(n)$, since there no more than n^3 cliques in a K_4 free graph.

Theorem 2.26. k-CLIQUE PARTITION, *for* $k \geq 3$, *is* **NP**-*complete for the class of graphs with no more than* n^k *maximal cliques.*

Proof. We can apply a simple transformation to reach from the case $k = 3$ to the case $k = 4$, and in general, from the case k to the case $k + 1$. For a graph $G = (V, E)$ which is an instance of the k-clique partition problem, we construct the graph $G = (V', E')$, where we obtain G' from G by adding $|V|/k$ vertices, each connected to every vertex of G.

Notice that a graph cannot be partitioned into disjoint k-cliques unless k divides $|V| = n$, and so, if this is not the case, we can simply output a trivial no instance, such as the one vertex graph, which cannot be partitioned into k-cliques for any $k > 1$. If on the other hand, k divides n, we construct V' and E' as

$$\begin{aligned} V' &= V \cup \{w_i : 1 \leq i \leq n/k\} \\ E' &= E \cup \{(w_i, v) : v \in V, 1 \leq i \leq n/k\}. \end{aligned}$$

This can clearly be done in polynomial time. Notice first that, as long as we do not output the trivial no instance, we know that $k + 1$ divides $|V'|$. We can prove that the reduction is correct by induction on k, the base case, for $k = 3$ is the triangle partition problem [21]. For $k \geq 3$, we build a graph G' using the transformation given above. If we take a k-clique partition for G, we can extend it to a $k + 1$ clique partition for G', as we must use n/k k-cliques to partition G, so we can take one of the new vertices into each k clique to form a $k + 1$ clique, as the new vertices are connected to all vertices in G. Similarly, for the other direction, any $k+1$ partition of G' must use one of the new vertices in each clique, as no two of the new vertices are adjacent. Also there are exactly n/k new vertices, and since there are $(n + n/k)/(k + 1) = n/k$ cliques in any $k + 1$ clique partition of G', so we will use all of the newly added vertices. Thus, we can simply remove the new vertices from the clique partition to obtain a k-clique partition of G. Thus, by induction on k, this reduction reduces the 3-clique partition problem to the k-clique partition problem for all $k \geq 3$. Notice also that at each step of the induction we increase the number of maximal cliques by a factor of n/k. Hence, the number of cliques in the final graph is bounded by $n^{k-3}p(n)$, if $p(n)$ is a bound on the number of cliques in the instance of the 3-clique partition problem, and since one possible choice for $p(n)$ is n^3, as the initial instance of the triangle partition problem was K_4 free, the constructed instance of the k-clique partition problem has no more than n^k cliques, which is polynomial in n, for fixed k. Thus, by the **NP**-completeness of the 3-clique partition problem on graphs with a polynomial bound on the number of cliques they may contain, we have the **NP**-hardness of the k-clique partition problem on the same graphs. We also have containment in **NP**, by the simple algorithm that guesses a k-clique partition and then verifies that the guessed partition does not include any vertex more than once, and that the guessed sets of size k are actually cliques. Thus, the problem of finding a k-clique partition is **NP**-complete for the class of graphs with fewer than n^k cliques. $\square$

2.5.3.5 Clique Partition and Clique Cover

Having shown the **NP**-completeness of k-CLIQUE PARTITION, we can easily show the **NP**-completeness of VERTEX CLIQUE PARTITION. This problem asks, for a graph G and an integer k, does G have a partition into k or fewer disjoint cliques? A formalization of this problem follows.

Problem. The VERTEX CLIQUE PARTITION problem is defined as:

Instance: A graph, $G = (V, E)$, and a natural number K.

Question: Can the vertices of graph G can be partitioned into k or fewer cliques?

In addition to this problem, we also have the closely related vertex and edge clique covering problems, which we will also consider in this section.

Problem. The VERTEX CLIQUE COVER problem is defined as:

Instance: A graph, $G = (V, E)$, and a natural number K.

Question: Is there a collection of cliques, $\mathcal{C} = \{C_1, C_2, \ldots, C_r\}$, with $r \leq k$, such that for each $v \in V$ there is some i such that $v \in C_i$?

Problem. The EDGE CLIQUE COVER problem is defined as:

Instance: A graph, $G = (V, E)$, and a natural number K.

Question: Is there collection of cliques, $\mathcal{C} = \{C_1, C_2, \ldots, C_r\}$, with $r \leq k$, such that for each $(u, v) \in E$ there is some i such that $u \in C_i$ and $v \in C_i$?

Notice that, by Theorem 2.5, the problems EDGE CLIQUE COVER and INTERSECTION NUMBER are effectively the same problem, since edge clique cover number $\theta_e(G)$ is equal to the intersection number. In this section we will refer to the problem as the EDGE CLIQUE COVER problem, as in this form it is more closely related to the other problems we consider here.

We can show that VERTEX CLIQUE PARTITION is **NP**-hard by reducing an n-vertex K_4 free instance, G, of 3-CLIQUE PARTITION to it. The reduction is trivial. The instance of VERTEX CLIQUE PARTITION is the graph G and the integer given by $\lfloor n/3 \rfloor$. We can partition the graph into at most $\lfloor n/3 \rfloor$ cliques if and only if we can partition the graph into triangles, since the graph is K_4 free. A potential complication arises on an instance where 3 does not divide n, but such an instance cannot be partitioned into triangles, so the reduction can output an arbitrary no instance of the clique partition problem. Once again, the reduction from the the exact cover by 3-sets problem to the 3-clique partition problem in [21] provides such instances of the problem which remain **NP**-complete, and yet K_4 free, which proves the **NP**-completeness of VERTEX CLIQUE PARTITION for the class of graphs with no more than n^3 cliques.

Once we have shown the clique partition problem to be **NP**-complete we can easily extend the result to VERTEX CLIQUE COVER problem. This is because the size of a minimum vertex clique cover is equal to the size of a minimum vertex clique partition, as given any minimum clique cover we can simply remove vertices which are covered by more than one clique from all but one of the cliques in the cover containing them. This process produces a clique partition that is no smaller than the minimum clique cover, and by observing that any clique partition is also a clique cover, we see that the size of a minimum vertex clique cover is equal to the size of a minimum vertex clique partition, and so the problem of finding a minimum cover or partition are clearly equivalent, on any class of graphs.

In addition to the **NP**-completeness of VERTEX CLIQUE COVER, we also have the **NP**-completeness of EDGE CLIQUE COVER for graphs with a polynomially bounded number of maximal cliques. This result comes directly from a result in [34], where the vertex clique cover problem is reduced to the edge clique cover problem by taking an instance G of VERTEX CLIQUE COVER, and adding $m + 1$ vertices to the graph that are adjacent to all the vertices of G, where m is the number of edges in G. If G has no more than $p(n)$ maximal cliques, then the resulting instance of the edge clique cover problem has no more than $(m + 1)p(n)$ maximal cliques. Thus, using this reduction, we have the **NP**-completeness of the edge clique cover problem, that is, the intersection number problem, for graphs with no more than $(m + 1)n^3$ maximal cliques.

2.5.3.6 Clique k-Partition

Finally, we encounter a problem that is solvable in polynomial time on graphs with few cliques. This problem is simply the k-COLOURABILITY problem in the complement graph, as any clique becomes an independent set, and we are seeking to partition the graph into cliques instead of independent sets, as done in the colouring problem. This problem is formally defined below.

Problem. The VERTEX CLIQUE k-PARTITION problem is defined as:

Instance: A graph, $G = (V, E)$.

Question: Can the vertices of graph G can be partitioned into k or fewer cliques?

Related to this problem are the problems of covering the vertices and edges of a graph with at most k cliques. These problems are defined here as well.

Problem. The VERTEX CLIQUE k-COVER problem is defined as:

Instance: A graph, $G = (V, E)$.

Question: Is there a collection of cliques, $\mathcal{C} = \{C_1, C_2, \ldots, C_r\}$, with $r \leq k$, such that for each $v \in V$ there is some i such that $v \in C_i$?

Problem. The EDGE CLIQUE k-COVER problem is defined as:

Instance: A graph, $G = (V, E)$.

Question: Is there a collection of cliques, $\mathcal{C} = \{C_1, C_2, \ldots, C_r\}$, with $r \leq k$, such that for each $(u, v) \in E$ there is some i such that $u \in C_i$ and $v \in C_i$?

A polynomial time algorithm for these problems for a graph G with no more than $p(n)$ maximal cliques is given in this section. The essential idea is that if we can cover a graph with cliques, we can also cover the graph with the same number of maximal cliques, by simply enlarging each of the cliques to some maximal clique, and so we can simply consider all collections of k maximal cliques.

For a graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, we first compute the set of all maximal cliques of G, $\mathcal{K}(G) = \{C_1, C_2, \ldots, C_r\}$, using the algorithm in [61], which runs in $O(nmr)$ time. Then we consider each of the $\binom{r}{k}$ sets of k maximal cliques of G. If any of these forms a clique cover, we output yes. If none of these are clique covers of G, we output no. Checking if each set is a clique cover can be performed in $O(n^2)$ time, for a total runtime of $O(n^2\binom{r}{k}) = O(n^2 p(n)^k)$ where $p(n)$ is a polynomial bound on the number of cliques, and so we have a polynomial time algorithm for the clique k-cover problem for graphs with a polynomially bounded number of maximal cliques.

Once again, by the equivalence of the VERTEX CLIQUE COVER and VERTEX CLIQUE PARTITION problems, this algorithm solves the k-partitioning problem as well, and since we can form a minimum edge k-clique cover from maximal cliques by using the same approach, we have a polynomial time algorithm, for any class of graphs with few cliques, for the edge k-clique cover problem. This problem is exactly the problem of determining for a graph G if there is an intersection representation for G of size at most k, for constant k.

No Text

Chapter 3

Overlap Representations

3.1 Introduction

Overlap representations have received considerably less study than intersection representations, even though overlapping is a very natural relation on pairs of sets. In order to discuss overlap representations formally, we present the following definition of an overlap representation, which is simply a formal version of the statement that an overlap representation of a graph is an assignment of sets to vertices such that vertices are adjacent in the graph if and only if the sets they are assigned intersect and neither set contains the other. This definition produces the interesting property that if we assign two vertices identical sets then they will have identical neighbourhoods, but will not be adjacent. This allows us to "clone" elements of an overlap representation, which can be useful for such vertices as the leaves of trees or members of the same partition of a complete bipartite graph.

Definition 3.1. Given a graph $G = (V, E)$, a collection $\mathcal{C} = \{S_v : v \in V\}$ is a p-overlap representation for G if for any $u, v \in V$ we have

$$(u, v) \in E \text{ if and only if } |S_u \cap S_v| \geq p, S_u \not\subseteq S_v, \text{ and } S_v \not\subseteq S_u.$$

We refer to a 1-overlap representation as simply an overlap representation. We also define the size of a representation as the number of elements in the union of all sets in the collection, which is

$$\left| \bigcup_{v \in V} S_v \right|,$$

and we let the *overlap number*, $\varphi(G)$, be the size of a minimum overlap representation for the graph G.

As can be seen from the definition, an overlap representation has many similarities to an intersection representation: sets assigned to adjacent vertices must intersect and disjoint

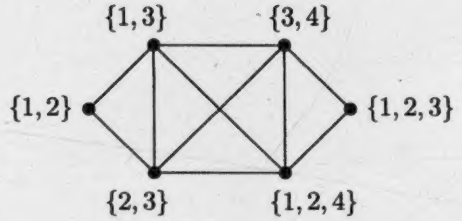

Figure 3.1: Example of a minimum overlap representation.

sets map to nonadjacent vertices. In the overlap case, however, the situation appears to be more complex, as not only do we need to ensure a stronger condition than intersection for adjacent vertices, we have a choice of representation, for every non-edge, of disjointedness or containment. As an example, consider the representation given in Figure 3.1, where there are nonadjacent vertices represented in each of these ways.

This added complexity causes the loss of many techniques that apply to intersection representations. Foremost among these losses is the notion of the edge clique graph, which was discussed in Section 2.1. In the case of an intersection representation, if we take an element of the representation and examine all those sets it is contained in, we find that the vertices associated with them form a clique. Doing the same in an overlap representation once again leaves us with a collection of vertices with intersecting sets, except here we may have non-edges represented by containment, and so, since the orientation implied by set containment forms a partial order, we can map elements of the representation to cocomparability graphs. Unfortunately, while covering all edges of a graph with cliques leads to an intersection representation, if we simply cover the edges of a graph with cocomparability graphs, we do not generally end up with a valid overlap representation. As an example, most cocomparability graphs do not have overlap number of one, even though they can be covered by a single cocomparability graph. This loss seems to indicate that no simple technique, such as vertex clique covering an edge clique graph, will yield a characterization of the problem of finding the minimum overlap representation. Nonetheless, a significant amount of structure appears to remain in an overlap representation, a structure that we will explore in the following sections.

Every graph has an overlap representation. To see this, we can take an intersection representation for a graph, and add a new element to each set. Sets will overlap in this representation if and only if the corresponding sets have nonempty intersection in the corresponding intersection representation. This construction also preserves the Helly property, and so all graphs have both overlap and Helly overlap representations.

One strategy for forming an overlap representation is to represent as many non-edges as possible using containment. This idea has some merit; we shall see in Section 3.4 that we can form an overlap representation of any cocomparability graph in such a way that all non-edges are represented by containment, and furthermore, this can be done using only a linear number of elements. In contrast, if we represent all non-edges by disjointedness, we obtain a restricted intersection representation, and there are graphs, as we have seen in Section 2.4, that require a quadratic sized intersection representation. While there does appear to be some benefit to making use of containment in an overlap representation, we cannot apply it blindly; for example, if we seek an overlap representation of the graph in Figure 3.1 where there are no disjoint sets, we find that we require one more element than the minimum representation given. One such minimum disjointedness-free representation can be obtained by replacing the set $\{3,4\}$ with the set $\{1,2,5\}$.

An interesting property of overlap representations is that the size of the representation required to cover a graph does not increase when the vertices of the graph are multiplied, where *vertex multiplication* is expanding a vertex into an independent set, such that the expanded vertices have the same adjacencies as the original vertex. This is stated formally as the following lemma, which we will use when constructing algorithms to find overlap representations.

Lemma 3.2. *If H can be obtained from G by vertex multiplication, then $\varphi(G) = \varphi(H)$.*

Proof. Since G is an induced subgraph of H, we immediately have $\varphi(G) \leq \varphi(H)$, since we can simply restrict a representation for H to the vertices of G.

In the other direction, let $\mathcal{C} = \{S_v : v \in V\}$ be an overlap representation for $G = (V, E)$. Each vertex v of G is mapped by vertex multiplication to an independent set (possibly of size one) in H. Let A_v be the set of all these vertices that represent v. Each vertex of H is in exactly one set A_v. We form the representation given by the following collection, for each vertex u of H,

$$\mathcal{D} = \{S'_u = S_v : u \in A_v\}.$$

Each vertex of A_v is assigned the same set, S_v, and so the sets associated with them do not overlap. For any two vertices w, z of H with $w \in A_u$ and $z \in A_v$, where $u \neq v$, the sets S'_w and S'_z overlap if and only if S_u and S_v do, which is exactly as required, since w and z are adjacent if and only if the vertices u and v they are expansions of are adjacent. Thus $\mathcal{D}$ is a representation for H of the same size as $\mathcal{C}$. In addition, since the sets of $\mathcal{D}$ interact in the same way that the sets of $\mathcal{C}$ do, $\mathcal{D}$ is a p-overlap representation or a representation with the Helly property if and only if $\mathcal{C}$ is. $\square$

Golumbic and Scheinerman established necessary and sufficient conditions in [27] for a class of graphs to be an overlap class of graphs of some family of nonempty sets. This characterization is very similar to the one given by Scheinerman [50] for intersection graphs. Any class of graphs that is hereditary, closed under vertex multiplication, and has a composition sequence is a class of overlap graphs of some family of sets. The only difference between this and the characterization of intersection graphs discussed in Section 2.1 is that vertex multiplication replaces vertex expansion. The reason for this is clear, in light of Lemma 3.2.

Unlike intersection graphs, where intersection representations of many families of sets have been studied, the only specific case of overlap representations that has received considerable attention are the circle graphs, which are usually defined as the intersection graphs of chords in a circle. These graphs can also be seen as the overlap graphs of intervals on a line, as is done by Gavril in [23]. To see how this transformation can be accomplished notice that, given any representation as intersecting chords, two chords intersect if and only if the arcs they subtend overlap. Notice also that in forming an arc from a chord, we may move around the circle in either direction, and either choice results in a valid overlap representation of arcs on a circle. This freedom allows us to make the final step, as if we choose the arcs in the representation in such a way that there is a portion of the circle that appears in no arc, then if we cut the circle at any point in this portion, we obtain a representation of the original circle graph as an overlap graph of intervals on this line. The algorithms in [23] for circle graphs operate essentially by exploiting this overlap representation. Čenek and Stewart [9] find a polynomial-time algorithm for the maximum independent set problem, given a general overlap representation and assuming that the weighted independent set problem for the associated class of intersection graphs is solvable in polynomial time. They also give a polynomial-time algorithm for the maximum clique problem, here with only the assumption that an overlap representation with the Helly property is given. Gavril, in [24], made similar observations for the more general classes of $\mathcal{G}$-mixed graphs, which, for a class of graphs G, are graphs that allow the edges to be decomposed into E_1 and E_2 such that the graph with edge set E_1 is in $\mathcal{G}$, the graph with edge set E_2 is a comparability graph, and these sets satisfy the additional property that if (w, v) is a directed edge in E_2, and $(u, v) \in E_1$ then $(u, w) \in E_1$ as well. A similar property also holds in the case of overlap graphs, where if $S_v \subseteq S_w$, and S_u overlaps S_v, then since S_u must also intersect S_w, if S_u does not overlap S_w, it must be the case that $S_u \subseteq S_w$. We will make extensive use of this property of overlap representations in the following sections.

3.2 Algorithms

In this section we examine some of the graphs for which minimum overlap representations, or the size of these representations, can be found. These relatively simple graphs are, at present, the only graphs that have known polynomial time algorithms to find minimum overlap representations. While these examples are not complex, the problems of finding minimum overlap and containment representations appear to be much harder than the problem of finding minimum intersection representations. As an example, it is simple to decide the extension problem for an intersection representation, as we have seen in Section 2.3.5, but we shall see in Sections 3.3.1 and 4.3.1 that this problem is **NP**-hard for overlap or containment representations.

In Section 3.2.1 we will study the size of an overlap representation for a clique by relating this problem to previous research in combinatorics. We will then see, in Section 3.2.2, how we can also apply these results to the case of complete k-partite graphs. These results will provide exact values for the size of a minimum representations of these graphs, but we delay the study of algorithms for computing these representations to Section 3.2.3, where we will prove some bounds on these values that will enable us to show that the algorithms we use are efficient. After examining these algorithms, we will change directions and study overlap representations of some simple graphs in Section 3.2.4. We will conclude the study of these algorithmic results in Section 3.2.5, where we will examine how we can take minimum overlap representations for the connected components of a graph, and use them to find a minimum representation for the whole graph.

3.2.1 Cliques

It should be noted that, as we seek to find a minimum overlap representation of a clique, we are concerned only with those overlap representations where no set contains any other. This is a severe restriction for an arbitrary graph, but in the case of a clique, as each pair of vertices is adjacent, any overlap representation must have this property. In the absence of containment, an overlap representation forms exactly a containment-free intersection representation. Thus, any overlap representation of a clique is also a containment-free intersection representation.

In order to find the minimum number of elements required to form an overlap representation for K_n, we can apply a theorem of Milner to find the maximum size of such a representation. We seek the maximum size of a collection of intersecting sets, none of which is contained in any other, or in the language of combinatorics, we seek the maximum size of an intersecting antichain. We introduce notation for this value, as we will meet it in several places.

Definition 3.3. The maximum size of a family, $\mathcal{C}$, of subsets of $\{1, 2, \ldots, m\}$ satisfying, for $p \geq 0$,

1. If $A, B \in \mathcal{C}$, with $A \neq B$, then $A \not\subseteq B$,

2. If $A, B \in \mathcal{C}$, then $|A \cap B| \geq p$,

is denoted $S(p, m)$.

The value of the function $S(p, m)$ is exactly the quantity given by Milner's Theorem, first published in 1966. This theorem is stated below.

Theorem 3.4 (Milner [39]).

$$S(p, m) = \binom{m}{\left\lfloor \frac{m+p+1}{2} \right\rfloor}$$

Some values of this function, for different values of m and p are given as Table 3.1. From the table it can be observed that this function grows quite quickly with m, an observation that we will make more formal in Section 3.2.3.1.

Also noted by Milner [39], is that it is easy to construct a collection that achieves this bound, by simply choosing all subsets of $\{1, 2, \ldots, m\}$ of size $\lfloor (m + p + 1)/2 \rfloor$. This, reformulated in the language of overlap representations, is precisely the content of the following corollary.

Corollary 3.5. *A minimum p-overlap representation for K_n has size given by*

$$\min \{m : n \leq S(p, m)\}$$

Proof. Consider any p-overlap representation, $\mathcal{B}$, of K_n. Any two elements of $\mathcal{B}$ must have intersection of size at least p, and no element can contain any other, as each pair of vertices in K_n forms an edge. Thus, we have satisfied the conditions of Definition 3.3, and so $|\mathcal{B}| \leq S(p, m)$, where $m = \left|\bigcup_{A \in \mathcal{B}} A\right|$.

	$S(0,m)$	$S(1,m)$	$S(2,m)$	$S(3,m)$	$S(4,m)$
$m=1$	1	1	0	0	0
2	2	1	1	0	0
3	3	3	1	1	0
4	6	4	4	1	1
5	10	10	5	5	1
6	20	15	15	6	6
7	35	35	21	21	7
8	70	56	56	28	28
9	126	126	84	84	36
10	252	210	210	120	120
11	462	462	330	330	165
12	924	792	792	495	495
13	1716	1716	1287	1287	715
14	3432	3003	3003	2002	2002
15	6435	6435	5005	5005	3003
16	12870	11440	11440	8008	8008
17	24310	24310	19448	19448	12376
18	48620	43758	43758	31824	31824
19	92378	92378	75582	75582	50388
20	184756	167960	167960	125970	125970

Table 3.1: Values of the function $S(p,m)$ for some values of m and p.

For any m, consider the collection given by $\mathcal{C}_m = \{A \subset \{1,2,\dots,m\} : |A| = \lfloor (m+p+1)/2 \rfloor\}$. As we have $\lfloor (m+p+1)/2 \rfloor \geq \lceil m/2 \rceil$, any two elements of $\mathcal{C}_m$ form an intersecting pair, and furthermore, no element is contained in any other, as they all have the same size. Counting the number of ways to form subsets of $\{1,2,\dots,m\}$, we obtain

$$|\mathcal{C}_m| = \binom{m}{\left\lfloor \frac{m+p+1}{2} \right\rfloor} = S(p,m).$$

Then, to find the minimum representation, we seek the minimum m that leaves enough room to form a p-overlap representation. This is because any p-overlap representation using m elements must be no larger than the collection $\mathcal{C}_m$, and so we can simply choose any n elements of $\mathcal{C}_m$ to obtain a p-overlap representation on the same number of elements. Thus, the size of the minimum p-overlap representation for K_n is given by the smallest m such that

$$n \leq |\mathcal{C}_m| = S(p,m),$$

as desired. $\square$

In addition to the p-overlap representation, we can demand a representation that satisfies yet another property. We can restrict our attention to those collections where any intersecting subcollection contains some common element. This is the Helly property, as given in Definition 1.3, and we refer to any p-overlap representation satisfying this property as a *Helly* p-overlap representation. It turns out that we can easily find the minimum size of a Helly p-overlap representation for a clique as well, using the same theorem of Milner, or in the case that $p = 0$, the theorem is usually known as Sperner's Theorem [51], the proof of which was initially published in 1928.

Proposition 3.6. *If C is a maximum collection of p-overlapping subsets of $\{1, 2, \ldots, m\}$ that satisfies the Helly property, then*

$$|C| = S(p-1, m-1).$$

Proof. Consider any collection C of p-overlapping subsets of $\{1, 2, \ldots, m\}$ that satisfies the Helly property. Notice that each set must have size at least two, since a one element set does not overlap any other set. Also, since each two elements of C overlap, they must intersect, and so by the Helly property, there must be an element in the intersection of all members of C. Without loss of generality, let this element be m. Consider the collection $\mathcal{B} = \{A \setminus \{m\} : A \in C\}$. As m is present in every element of C, and every element of C has at least two elements, this does not create any duplicate sets in $\mathcal{B}$ that were not already duplicated in C. If any element of $\mathcal{B}$ is contained in any other, then there are two sets, in C, that do not overlap, which contradicts the definition of C. Similarly, if there are any two elements in $\mathcal{B}$ that do not have a common intersection of size at least $p-1$, then we can find sets in C that do not p-overlap. Thus, we have constructed $\mathcal{B}$, which is a $p-1$-overlapping collection of subsets of $\{1, 2, \ldots, m-1\}$, and so we have

$$|C| = |B| \leq S(p-1, m-1).$$

To see that this bound can be achieved, we can simply take a $p-1$-overlapping collection of subsets of $\{1, 2, \ldots, m-1\}$ of size $S(p-1, m-1)$, which must exist by Milner's Theorem. If we add the element m to each set in this collection, we have a p-overlapping collection of size $S(p-1, m-1)$ that satisfies the Helly property. ☐

Once again, this can be immediately applied to find the minimum number of elements needed to form a representation for K_n.

Corollary 3.7. *A minimum Helly p-overlap representation for K_n has size given by*

$$\min\{m : n \leq S(p-1, m-1)\}$$

3.2.2 Complete k-Partite Graphs

Using the ideas developed in the previous section, we can immediately find the minimum number of elements needed to form overlap representations of the complete k-partite graphs. It is interesting that while the minimum number of elements needed to form a k-overlap representation for a complete bipartite graph is not difficult to find, the problem of finding the number of elements needed for a k-intersection representation appears to be much more difficult [15, 18]. It should also be noted that here the connection to a containment-free intersection representation does not hold. Our overlap representation, which can be derived from the proof of Lemma 3.2, will make use of containment, albeit in a rather simple way, and so it will not form a containment-free intersection representation.

Proposition 3.8. *Let* $G = (X_1 \cup X_2 \cup \cdots \cup X_k, E)$ *be a complete k-partite graph. If* $\mathcal{C}$ *is a minimum p-overlap representation of G, then the number of elements used by* $\mathcal{C}$ *is the number of elements required to form a minimum p-overlap representation of* K_k*. Furthermore, a minimum Helly p-overlap representation uses the same number of elements as a minimum Helly p-overlap representation for* K_k*.*

Proof. Since G can be obtained from K_k by vertex multiplication, we can apply Lemma 3.2 to show that a minimum overlap representation for G has the same size as a minimum overlap representation for K_k. By the remarks at the end of the proof of this lemma, this construction also preserves the Helly property, and representations with overlap of size p. □

By observing that the graph O_t is simply a complete t-partite graph, with two vertices in each partition, we can find an optimal p-overlap representation for this graph as well, by the preceding proposition.

3.2.3 Computing a Representation of K_n

In this section we investigate some computational issues involved in finding overlap representations of cliques and k-partite graphs. This is done by first finding bounds on the quantity $S(p, n)$ in the size of the graph, in Section 3.2.3.1. Using these bounds we examine, in Section 3.2.3.2, the complexity of computing the value of $S(n, m)$ that determines how many elements are in the minimum overlap representation for a clique on n vertices, or an n-partite complete graph. Finally, using these results, as well as the constructive proofs of Section 3.2.1, we give, in Section 3.2.3.3, a simple algorithm to produce a minimum overlap representation of K_n.

3.2.3.1 Bounds in the Size of the Graph

In order to gain a more useful view of the size of the required representation for a given graph, we seek to unwind the expression

$$\min\left\{m : n \leq \binom{m}{\lfloor \frac{m+k+1}{2} \rfloor}\right\},$$

to obtain a more useful bound on m in terms of n and k. In order to do this, we give up precision, obtaining asymptotically tight bounds. We will make heavy use of Stirling's Approximation, which can be found in [4] as well as many other places,

$$\sqrt{2\pi n}(n/e)^n \leq n! \leq e^{1/(12n)}\sqrt{2\pi n}(n/e)^n. \tag{3.1}$$

Armed with this identity, we have the following simple corollary, which is obtained by simply substituting the above into the expansion of the binomial coefficient.

Lemma 3.9. *For* $1 \leq k < n$,

$$\binom{n}{k} \geq \sqrt{\frac{1}{8\pi k}}\left(\frac{n}{k}\right)^k\left(\frac{n}{n-k}\right)^{n-k}.$$

Proof. By simple, but tedious, expansion, using Equation (3.1) we have

$$\begin{aligned}
\binom{n}{k} = \frac{n!}{k!(n-k)!} &\geq \frac{\sqrt{2\pi n}(n/e)^n}{2\pi e^{1/(12k)+1/(12(n-k))}\sqrt{k(n-k)}(k/e)^k((n-k)/e)^{n-k}} \\
&\geq \frac{1}{e^{1/6}}\sqrt{\frac{n}{2\pi k(n-k)}}\frac{n^n}{k^k(n-k)^(n-k)} \\
&\geq \frac{1}{2}\sqrt{\frac{n}{2\pi k(n-k)}}\left(\frac{n}{k}\right)^k\left(\frac{n}{n-k}\right)^{n-k} \\
&\geq \sqrt{\frac{n-k}{8\pi k(n-k)}}\left(\frac{n}{k}\right)^k\left(\frac{n}{n-k}\right)^{n-k} \\
&= \sqrt{\frac{1}{8\pi k}}\left(\frac{n}{k}\right)^k\left(\frac{n}{n-k}\right)^{n-k}
\end{aligned}$$

as in the statement of the lemma. ☐

Using this lemma, we can bound the size of the minimum overlap representation of the graphs we have considered. The proof here is simply a calculation.

Theorem 3.10. *Let* $0 \leq k < n$, *with* k *not depending on* n, *and let* m *be the size of a* k*-overlap representation for a complete graph on* n *vertices, which is given by*

$$m = \min\left\{x : n \leq \binom{x}{\lfloor \frac{x+k+1}{2} \rfloor} = S(k,x)\right\}.$$

Then we have $m \in \Theta(k + \log n)$.

Proof. We can show a simple lower bound by noticing that there are only 2^m potential subsets of $\{1, 2, \ldots, m\}$, and so we must have $n < 2^m$, which implies that $m \in \Omega(\log n)$. We will show a better lower bound later, but we will need this crude one to notice that as we consider large values of n, we also must consider large values of m.

Turning to an upper bound, notice that, by the definition of m, we have

$$\binom{m-1}{\lfloor \frac{m+k}{2} \rfloor} < n \leq \binom{m}{\lfloor \frac{m+k+1}{2} \rfloor}. \tag{3.2}$$

Using this, and Lemma 3.9, we have

$$\begin{aligned} n &> \binom{m-1}{\lfloor \frac{m+k}{2} \rfloor} \\ &\geq \binom{m-1}{\frac{m+k}{2}} \\ &\geq \sqrt{\frac{2}{8\pi(m+k)}} \left(\frac{2(m-1)}{m+k}\right)^{(m+k)/2} \left(\frac{2(m-1)}{m-k}\right)^{(m-k)/2} \\ &\geq 2^m \sqrt{\frac{1}{4\pi(m+k)}} \left(\frac{m-1}{m+k}\right)^{(m+k)/2} \left(\frac{m-1}{m-k}\right)^{(m-k)/2}; \end{aligned} \tag{3.3}$$

and focusing on the final two terms, we further obtain

$$\begin{aligned} \left(\frac{m-1}{m+k}\right)^{(m+k)/2} \left(\frac{m-1}{m-k}\right)^{(m-k)/2} &= \frac{(m-1)^m}{(m+k)^k \left((m+k)(m-k)\right)^{(m-k)/2}} \\ &= \frac{(m-1)^m}{(m+k)^k (m^2-k^2)^{(m-k)/2}} \\ &\geq \frac{(m-1)^m}{(m+k)^k (m^2)^{(m-k)/2}} \\ &= \frac{(m-1)^m}{(m+k)^k m^{m-k}} \\ &\geq \frac{(m-1)^m m^k}{(2m)^k m^m} \\ &\geq 2^{-k} \left(\frac{m-1}{m}\right)^m. \end{aligned} \tag{3.4}$$

Then, since we are interested in the asymptotic behaviour of m, if we take n large enough to require that $m \geq 4$, and noticing that $\log((m-1)/m) > \log(3/4) > -1/2$, we may substitute (3.4) into (3.3), and take logarithms to obtain

$$\begin{aligned}
\log n &> \log\left(2^m\sqrt{\frac{1}{4\pi(m+k)}}2^{-k}\left(\frac{m-1}{m}\right)^m\right) \\
&= m-k-\frac{\log(4\pi(m+k))}{2}+m\log\left(\frac{m-1}{m}\right) \\
&> m-k-\frac{\log(4\pi(m+k))}{2}-\frac{m}{2} \\
&= \frac{m}{2}-k-\frac{\log(4\pi(m+k))}{2} \\
&\geq \frac{m}{2}-k-\frac{\log 4\pi}{2}-\log(mk) \\
&= \frac{m}{2}-k-\frac{\log 4\pi}{2}-\log m-\log k.
\end{aligned}$$

The second last line is due to the fact that, for $m, k \geq 1$, we have $m+k < 2mk$. These last two lines are not valid for $k=0$, but in this case we can reach the same conclusion, without the $\log k$ term, and the remainder of the analysis will be the same, since the $\log k$ term is not significant. Once more, we can take n large enough to require m large enough so that $\log m < m/2$, we then have, by the above, and setting $C = \log(4\pi)/2$,

$$\log n > (m/2)-k-(m/4)-\log k-C = (m/4)-k-\log k-C,$$

which is,

$$(m/4) < k+\log n+\log k+C,$$

and so we have $m \in O(k+\log n)$, as desired.

A lower bound, tighter than the previous lower bound, can be demonstrated more easily. Using Stirling's Approximation, given as Equation (3.1), we have the observation, which can be found in [4], that

$$\binom{n}{k} \leq \frac{n^k}{k!} \leq \frac{1}{2\sqrt{k}}\left(\frac{en}{k}\right)^k \leq \left(\frac{en}{k}\right)^k.$$

We can immediately apply this to the task at hand, obtaining

$$\begin{aligned}
n &\leq \binom{m}{\lfloor\frac{m+k+1}{2}\rfloor} \\
&\leq \binom{m}{\frac{m+k}{2}} \\
&= \binom{m}{\frac{m-k}{2}} \\
&\leq \left(\frac{2em}{m-k}\right)^{(m-k)/2} \\
&\leq (2e)^{(m-k)/2}\left(\frac{m}{m-k}\right)^{(m-k)/2}.
\end{aligned}$$

Notice then that, by the previous weaker lower bound, we can take n large enough to require m large enough so that $m - k > m/2$. Applying this, we have

$$n \leq (2e)^{(m-k)/2} \left(\frac{m}{m-k}\right)^{(m-k)/2} \leq (2e)^{(m-k)/2} \left(\frac{m}{m/2}\right)^{(m-k)/2} = (4e)^{(m-k)/2},$$

which, taking logarithms, gives

$$\log n + \frac{k}{2}\log(4e) \leq \frac{m}{2}\log(4e).$$

This immediately implies the desired bound of $m \in \Omega(k + \log n)$. □

3.2.3.2 Calculating S(p,m)

We are now in position to build an efficient algorithm to find, for a given n, the minimum m such that $n \leq S(p,m)$. We will do this in a straightforward way, by computing values of $S(p,k)$ for increasing values of k, until we find a value that is larger than n. We will be able to use the bounds of Section 3.2.3.1 to show that this process does not generate too many intermediate values, as well as some dynamic programming to speed up the computation of $S(p,k)$. This will turn out to be simpler, and as efficient as the alternate strategy of using binary search to find the minimum m. In order to demonstrate this algorithm, we will first show how to compute $S(p,m)$ in terms of $S(p,m-1)$. To do this, we will need to make use of the well-known identity

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}, \tag{3.5}$$

which can be found, for example, in [33]. Using this, we can directly show the desired relationship.

Proposition 3.11. *Let* $0 \leq p < m$*, then, if* $m + p$ *is even,*

$$S(p,m) = \frac{2m}{m-p} S(p,m-1),$$

and if $m + p$ *is odd,*

$$S(p,m) = \frac{2m}{m+p+1} S(p,m-1).$$

Proof. If $m + p$ is even, we have

$$\begin{aligned}
\frac{2m}{m-p} S(p,m-1) &= S(p,m-1) + \frac{m+p}{m-p} S(p,m-1) \\
&= S(p,m-1) + \frac{m+p}{m-p}\binom{m-1}{(m+p)/2} \\
&= S(p,m-1) + \frac{(m+p)}{(m-p)} \frac{(m-1)!}{(m-1-(m+p)/2)!((m+p)/2)!} \\
&= S(p,m-1) + \frac{(m+p)}{(m-p)} \frac{(m-1)!}{((m-p)/2-1)!((m+p)/2)!},
\end{aligned}$$

at which point we can divide both the $(m+p)$ and $(m-p)$ terms by two, and apply them to the factorials in the denominator of the final term, obtaining

$$\begin{aligned}
\frac{2m}{m-p}S(p,m-1) &= S(p,m-1) + \frac{(m-1)!}{((m-p)/2)!((m+p)/2-1)!} \\
&= \binom{m-1}{(m+p)/2} + \binom{m-1}{(m+p)/2-1}.
\end{aligned}$$

Finally, we apply Equation (3.5) to this to show that

$$\frac{2m}{m-p}S(p,m-1) = \binom{m-1}{(m+p)/2} + \binom{m-1}{(m+p)/2-1}. = \binom{m}{(m+p)/2} = S(p,m).$$

The other case, when $m+p$ is odd, is similar, and so, proceeding in the same manner, we have

$$\begin{aligned}
\frac{2m}{m+p+1}S(p,m-1) &= S(p,m-1) + \frac{m-p-1}{m+p+1}S(p,m-1) \\
&= S(p,m-1) + \frac{m-p-1}{m+p+1}\binom{m-1}{(m+p-1)/2},
\end{aligned}$$

and once again, we can divide each of the terms $m-p-1$ and $m+p+1$ by two, and work them into the expansion of the binomial coefficient, exactly as in the even case. Doing this, we obtain

$$\begin{aligned}
\frac{2m}{m+p+1}S(p,m-1) &= S(p,m-1) + \frac{(m-1)!}{((m-p-1)/2-1)!((m+p+1)/2)!}, \\
&= \binom{m-1}{(m+p+1)/2-1} + \binom{m-1}{(m+p+1)/2},
\end{aligned}$$

which, again by Equation (3.5), shows that

$$\frac{2m}{m+p+1}S(p,m-1) = \binom{m}{(m+p+1)/2} = S(p,m),$$

as required. □

Using this relation, we can compute the smallest m such that $n \leq S(p,m)$ by simply starting with $S(p,p)$, as it is clear that for sets to have intersection of size p, there must be at least p elements in the union of all the sets. The value of $S(p,p)$ is also trivial to compute, as we have

$$S(p,p) = \binom{p}{\lfloor (p+p+1)/2 \rfloor} = \binom{p}{p} = 1,$$

where we assume $p \geq 1$. If $p = 0$ we use $S(0,1) = 1$ as the base for the recurrence. Thus, for a given n, to find the size of the smallest overlap representation for K_n we can start with $S(p,p) = 1$ or $S(0,1) = 1$, and then, by Proposition 3.11 we can find, given $S(p,m)$, the value of $S(p,m+1)$, we continue in this way until we find m such that $n \leq S(p,m)$.

By Theorem 3.10, this process must terminate, as there is an upper bound on the size of the required m. Furthermore, by this theorem, we require the computation of only $\Theta(p + \log n)$ successive values of $S(p, m)$. Treating p as a constant, this produces an algorithm with runtime in $O(\log n)$, in the unit-cost model, which we feel is reasonable in this case, as the size of m is only logarithmic in n, and so is likely to fit within the word size of the machine performing the computation. This produces a polynomial time algorithm, given n, to find the size of a minimum intersection representation of O_n, and the size of a minimum p-overlap representation for K_n on a complete n-partite graph.

3.2.3.3 Finding a Representation

We have now assembled all of the required pieces to compute a minimum p-overlap representation for a clique on n vertices, a p-overlap representation of a complete n-partite graph, or an intersection representation for the graph O_n, which was examined in Section 2.3.4.

An algorithm to find a p-overlap representation of K_n begins by computing m, the minimum size that such a representation must have, which is the smallest m such that $n \leq S(p, m)$. This is done in $O(\log n)$ time, by iteratively applying the recurrence in Proposition 3.11, as discussed in Section 3.2.3.2. Once this value has been calculated, to find a minimum representation for K_n, as noted in the proof of Corollary 3.5, we can simply take any n of the subsets of $\{1, 2, \ldots, m\}$ of cardinality $(m + p + 1)/2$. Since $m \in O(\log n)$, n of these subsets can be found in $O(n)$ time, by constructing recursively all subsets of $\{1, 2, \ldots, m\}$ while keeping track of the number of elements in each set, so that only those sets of the correct cardinality are produced as output. Thus, we can find, in linear time, a p-overlap representation for K_n.

In order to find a p-overlap representation of K_n that satisfies the Helly property, we may do essentially the same thing. Here we once again compute the minimum size that such a representation must have, which is the smallest m such that $n \leq S(p - 1, n - 1)$. We can then find a $(p - 1)$-overlap representation for K_n of this size, before adding a new element to each set of the representation. This can be done, using the previous algorithm, in $O(n)$ time, and the representation produces is minimum, by Corollary 3.7.

These algorithms can also be immediately extended to find representation for complete n-partite graphs, as a minimum representation for such a graph is obtained by simply finding a minimum representation for K_n, and assigning to the vertices of the ith partition of the graph the set assigned to the ith vertex of K_n. This produces, in $O(n)$ time, an optimal representation, by Proposition 3.8.

In addition to this, as noted in Section 2.3.4, we can use the algorithm to find a p-overlap representation for a clique K_{n-1} to find an intersection representation for O_n. First we find a 1-overlap representation for K_{n-1}, which is simply an overlap representation. If the sets used in this overlap representation are subsets of $\{1, 2, \ldots, m-1\}$, we instead view them as subsets of $\{1, 2, \ldots, m\}$, which reverses the transformation applied in Lemma 2.14. Finally, we apply the construction from the proof of Lemma 2.13, building m maximal cliques, which will each contain one vertex of each nonadjacent pair, as specified in the lemma. These cliques will form a minimum edge-clique-cover for O_n, by taking, for each vertex, the set of all cliques in the cover that contain it, we have a minimum intersection representation, as discussed in Section 2.3.1. The time used for these constructions is $O(n \log n)$, as we must construct $O(\log n)$ cliques, each of linear size.

3.2.4 Paths, Cycles, and Caterpillars

The problem of finding a minimum overlap representation for a path would at first appear to be a trivial problem. A minimum intersection representation is simple to find, as there is only one possible edge-clique-cover, the one consisting of each maximal clique, which is simply each edge on the path. While it is essentially no harder to find an overlap representation of a path, proving the optimality of the representation is more difficult. Once we have shown the size that an overlap representation of a path must have, we can immediately apply the result, with a little bit of work, to the case of cycles and caterpillars. In order to prove such a lower bound, we make use of a simple observation, which is given as the following lemma.

Lemma 3.12. *If A, B, C are three sets such that $A \subseteq C$, A overlaps B, but B does not overlap C then $B \subseteq C$.*

Proof. Since A and B overlap, B has nonempty intersection with C. If B does not overlap C then either $C \subseteq B$ or $B \subseteq C$. If $A \subseteq C \subseteq B$, then A and B must not overlap, which is a contradiction that shows that $B \subseteq C$. □

We can amplify this lemma to the following stronger lemma that we will use to argue bounds on the size of an overlap representation for a graph based on the size representations for the components of the graph, in Section 3.2.5.

Lemma 3.13. *Let $G = (V, E)$ be a graph, $X, Y \subseteq V$ such that $G[X]$ and $G[Y]$ are connected induced subgraphs of G such that there are no edges $(x, y) \in E$ with $x \in X$ and*

$\in Y$, and let $\mathcal{C} = \{S_v : v \in V\}$ be an overlap representation for G. Let $U = \bigcup_{x \in X} S_x$ be the set of all elements in the representation for X. If for some $x \in X$ and some $y \in Y$, $S_x \subseteq S_y$ then for any $v \in Y$ either $U \subseteq S_v$ or $U \cap S_v = \emptyset$. Furthermore, if this is the case, then no set S_x for $x \in X$ contains any set S_y for $y \in Y$.

Proof. Let $x_1 \in X$ and $y \in Y$ such that $S_{x_1} \subseteq S_y$. We will first show that $U \subseteq S_y$. To see this, let $x_k \in X$, and let $x_1, x_2, \dots, x_k$ be a path from x_1 to x_k in $G[X]$. We then have $S_{x_1} \subseteq S_y$, S_{x_2} overlaps S_{x_1}, and S_{x_2} does not overlap S_y since x_2 and y are not adjacent. Applying Lemma 3.12, we see that $S_{x_2} \subseteq S_v$. This argument can then be repeated, with x_2 in place of x_1, since these vertices are both in $G[X]$, and so we also have $S_{x_3} \in S_y$. Continuing in the fashion, it must be the case that $S_{x_k} \subseteq S_y$, and as $x_k \in X$ was arbitrary, we have $U \subseteq S_y$, as desired.

To complete the proof of the theorem, notice that $y \in Y$ was chosen to be an arbitrary element of Y such that there is some $x \in X$ with $S_x \subseteq S_y$. Let $x \in X$ and $v \in Y$ be two vertices such that S_x and S_v intersect. We will show that $S_x \subseteq S_v$. Since x and v are not adjacent, we need only to rule out the case that $S_v \subseteq S_x$. If this is the case, then by applying the previous argument with X and Y reversed, S_x contains $S_{y'}$ for all $y' \in Y$. In particular, S_x contains the set S_y that contains U, the set of all elements used in the representation of $G[X]$. This implies that $U \subseteq S_y \subseteq S_x$, and so S_x contains all other sets $S_{x'}$ for $x \in X$, which is a contradiction, since $G[X]$ is connected. Thus we have shown the last sentence of the theorem, and also that any set S_v for $v \in Y$ is either disjoint from all S_x for $x \in X$, or there is some $x \in X$ such that $S_x \subseteq S_v$, which implies by the previous argument that $U \subseteq S_v$. □

This lemma, while powerful, is rather awkward to use. In this section we will use the following simplification of it to argue bounds on the overlap number for paths, cycles, and caterpillars.

Lemma 3.14. *Let $G = (V, E)$ be a graph, and let $\mathcal{C} = \{S_v : v \in V\}$ be an overlap representation of G. Fix $v \in V$, and let, for $u \in V \setminus N[v]$, $A_v(u)$ be the vertex set of the connected component of $G[V \setminus N[v]]$ that contains u. If $S_u \subseteq S_v$, then*

$$\bigcup_{w \in A_v(u)} S_w \subseteq S_v.$$

Proof. Let $X = A_v(u)$ and $Y = \{v\}$. Applying Lemma 3.13 gives

$$\bigcup_{w \in A_v(u)} S_w = U \subseteq S_v,$$

since $S_u \subseteq S_v$, so S_v is not disjoint from U. □

With this lemma in hand, we can show a lower bound on the size of an overlap representation for the graph P_n, the path on n vertices. This lower bound will be matched by a simple construction, given as part of the proof, that can be immediately transformed into a efficient algorithm for generating an overlap representation of P_n. This theorem does not hold for P_2, as $\varphi(P_2) = 3$.

Theorem 3.15. *For $n \geq 3$, we have $\varphi(P_n) = n$.*

Proof. For $n = 3$, we observe that $\{\{1,2\},\{2,3\},\{1,2\}\}$ is a minimum overlap representation, since we need at least three elements to represent a single edge. For $n \geq 4$, we label the vertices of P_n from the set $\{1,2,\ldots,n\}$ in increasing order, and let $C = \{S_1, S_2, \ldots, S_n\}$ be a minimum overlap representation for P_n, with vertex i corresponding to the set S_i. We can then observe that one of two cases must hold: either S_1 contains none of $\{S_3, S_4, \ldots, S_n\}$, or it contains each S_i for $i \geq 3$. This is due to the fact that if S_1 contains S_i for any $i \geq 3$, then it must contain S_k for all $3 \leq k \leq n$, by Lemma 3.14, since these are exactly the vertices of $P_n[\{3,4,\ldots,n\}]$. Notice also that these two cases collapse, as if S_1 contains all S_i for $i \geq 3$, then in particular, $S_n \subseteq S_1$, and so, if we consider the reversal of the path, we find that S_n contains none of the other sets, as $n \geq 4$, which forbids the case that $S_1 = S_n$, and so we need only consider the first case.

To this end, let the representation, without loss of generality, be such that S_1 contains none of $\{S_3, \ldots S_n\}$. Notice that with the exception of S_2, the elements of S_1 are either all contained in one of the other sets, or none of them are. We will form a representation for P_{n-1} where these elements are compressed into a single element. We consider the collection given by $C' = \{S_2 \cup S_1, S_3, \ldots S_n\}$. As S_1 and S_2 share at least one element, this enlarging of S_2 will not cause S_k to overlap $S_2 \cup S_1$ for any $k \geq 4$. To see this, we consider two cases. The first case is that $S_1 \subseteq S_k$, but then, by Lemma 3.12, we have $S_2 \subseteq S_k$ as well, which implies that $S_1 \cup S_2 \subseteq S_k$, as desired. In the other case we have S_1 disjoint from S_k, but in this case we can observe that $S_2 \not\subseteq S_k$, as this would imply, again by Lemma 3.12, that $S_1 \subseteq S_k$. Since $S_2 \not\subseteq S_k$, it is either disjoint from S_k, in which case $S_1 \cup S_2$ is as well, or $S_k \subseteq S_2$, which implies that $S_k \subseteq S_1 \cup S_2$, as required. Similarly, in the collection $C'' = \{S_2 \setminus S_1, S_3, \ldots, S_n\}$, the set $S_2 \setminus S_1$ will not to overlap any set S_k for $k \geq 4$. To see this, notice that, since we reduce the size of S_2, and S_1 and S_2 overlap, so $|S_2 \setminus S_1| > 0$, the only troublesome case is when $S_k \subseteq S_2$. If $S_2 \setminus S_1$ overlaps S_k, but $S_k \subseteq S_2$, then S_1 and S_k have nonempty intersection, which forces $S_1 \subseteq S_k$.

Applying Lemma 3.12 we see that $S_2 \subseteq S_k$ as well, which implies that $S_2 = S_k$, which is a contradiction as S_2 overlaps S_1, but S_k does not. Thus the set $S_2 \setminus S_1$ must not overlap S_k for any $k \geq 4$.

Thus, we need only verify that one of these two collections preserves the overlap between S_3 and the replacement for S_2. To see that at least one suffices, let C' fail to be an overlap representation for P_{n-1}, which implies that $S_3 \subseteq S_2 \cup S_1$, as we have only enlarged S_2. This implies that $S_1 \cap S_3 \neq \emptyset$, as S_3 is contained in neither S_1 or S_2, but it is contained in their union. Then, since S_1 does not contain any other set in the representation, we must have $S_1 \subseteq S_3$ as these two vertices are not adjacent in the path. Notice also that, since S_3 is contained in $S_1 \cup S_2$, and S_3 is not contained in S_1, we must have $S_3 \cap (S_2 \setminus S_1) \neq \emptyset$. Seeking a contradiction, we assume that S_3 and $S_2 \setminus S_1$ also do not overlap. Since C is an overlap representation for P_n, we must have $S_3 \not\subseteq S_2 \setminus S_1 \subseteq S_2$, as the vertices associated with S_2 and S_3 are adjacent. This leaves only one way for S_3 to fail to overlap $S_2 \setminus S_1$, which is $(S_2 \setminus S_1) \subseteq S_3$. If this is the case, then we have $S_2 \subseteq S_3$, as we know that $S_1 \subseteq S_3$, which we derived from the failure of C'. This contradicts the fact that C is an overlap representation for P_n, and so we have shown that if C' is not a valid representation for P_{n-1}, then C'' must be. Notice then that in both of these representations, each set either contains S_1 or is disjoint from it, and so there is no loss in replacing the elements of S_1 with a single element. This reduces the size of the representation by at least one, as a set needs at least two elements to overlap another set. Hence, we have formed a representation for P_{n-1} from a representation of P_n, and this new representation has size at least one less than the old. By induction on n, we have shown that

$$\left| \bigcup_{S_k \in C} S_k \right| = \varphi(P_n) \geq 1 + \varphi(P_{n-1}) = 1 + n - 1 = n.$$

This demonstrates that $\varphi(P_n) \geq n$, and so to show the desired equality, it is sufficient to build a representation of this size. This is simple to do. Consider the representation for P_n given by, for $1 \leq i \leq n-1$,

$$\begin{aligned} S_i &= \{i, i+1\} \\ S_n &= \{1, 2, \ldots, n-1\}. \end{aligned}$$

Notice that in this representation, on the first $n-1$ vertices, the set S_i overlaps only the sets S_{i-1} and S_{i+1} and is disjoint from the other sets, with the exception of S_n. Also, S_n contains all sets except S_{n-1}, which it overlaps, and so this is an overlap representation for P_n using n elements. This proves that $\varphi(P_n) = n$, as desired. □

The representation used in the proof of the theorem is optimal in the number of elements used, and can be constructed in $O(n)$ time, which is asymptotically optimal, as a representation needs to have linear size. Thus we can view this construction as an efficient algorithm to find an overlap representation of a path.

Having found the overlap number of a path, we can find immediate lower bounds on the size of the overlap representation for some other simple graphs. The first of these is C_n, the cycle on n vertices. Once again, the lower bound will be matched by a simple construction, which can be transformed immediately into an algorithm with running time linear in n. This theorem will not hold for C_3, as we have $\varphi(C_3) = 3$, by taking for the representation the three sets that can be formed by choosing two of three elements.

Corollary 3.16. *For* $n \geq 4$, *we have* $\varphi(C_n) = n - 1$.

Proof. To see that $\varphi(C_n) \geq n-1$ we simply observe that by Theorem 3.15, the size of the representation for any $n-1$ of the n vertices is at least $n-1$, and so it remains only to construct a representation using $n-1$ elements. We do this by setting, for $1 \leq i \leq n-2$,

$$S_i = \{i, i+1\},$$

which forms an overlap representation for a path of $n-2$ vertices, using $n-1$ elements. We add to this representation $S_{n-1} = \{1, 2, 3, \ldots, n-2\}$ and $S_n = \{2, 3, 4, \ldots, , n-1\}$, noting that S_{n-1} overlaps only S_n and S_{n-2}, containing the other sets, and that S_n overlaps only S_{n-1} and S_1 as it contains all other sets in the collection. Thus, the collection $\mathcal{C} = \{S_1, S_2, \ldots, S_n\}$ forms an overlap representation for C_n using $n-1$ elements, proving that $\varphi(C_n) = n-1$. ☐

The next simple class of graphs we can find a minimum representation for are the caterpillars, which are trees with a very simple structure. A definition of these graphs, equivalent to the one given below, can be found in [64].

Definition 3.17. A tree $T = (V, E)$ is a *caterpillar* if the non-leaf vertices form a path, known as the *spine* of the caterpillar.

With the definition in hand, we can use Theorem 3.15 to find a lower bound on the size of an overlap representation for a caterpillar. We will once again be able to pair this with a simple construction to show that the bound is tight. This construction will then lead to an efficient algorithm for finding an overlap representation.

Corollary 3.18. *For $T = (V, E)$ a caterpillar with spine containing $k \geq 1$ vertices, we have $\varphi(T) = k + 2$.*

Proof. We will show that the size of a minimum overlap representation for a caterpillar has size determined by the size of the overlap representation for the longest path in the caterpillar. Let T be a caterpillar, and label the vertices of the spine in order $\{1, 2, \dots, k\}$, and let L_i be the leaves connected to vertex i of the spine. Notice that any longest path in T has a vertex in L_1 and a vertex in L_k as endpoints, with the remaining vertices being those of the spine. This allows the above labelling scheme to be implemented in linear time, as a longest path in a tree can be found in linear time. Also notice that the longest path in T contains $k + 2$ vertices, and so Theorem 3.15 provides a lower bound of $\varphi(T) \geq k + 2$.

To show a tight bound, we need only find a representation of the correct size. The representation used is very similar to the one used in the proof of Theorem 3.15. For T a caterpillar, with nodes labelled $1, 2, \dots k$ that form the spine, with node i adjacent to nodes $i - 1$ and $i + 1$, and L_i the set of leaves adjacent to vertex i, consider the representation given by, for $1 \leq i \leq k$,

$$\begin{aligned} S_i &= \{i+1, i+2\} \\ S_{L_i} &= \{1, 2, \dots, i+1\}, \end{aligned}$$

where the set S_{L_i} is associated with all vertices in L_i. This representation coincides with the one previously given for paths, since viewing a path on n vertices as a caterpillar produces a caterpillar with $n - 2$ vertices on the spine, and two leaves, one on each end of the path. To see that the given representation is correct, notice that two vertices of the spine i and j overlap if and only if $|i - j| = 1$. Notice also that the sets assigned to two leaves will never overlap, as S_{L_i} overlaps all S_{L_j} for $j \leq i$. In addition, S_{L_i} overlaps only S_i, since S_{L_i} contains S_j for $j < i$, and S_{L_i} is disjoint from S_j for $j > i$. Thus we have constructed an overlap representation for T using $k + 2$ elements. This proves that for any caterpillar T with $k \geq 1$ vertices on the spine, $\varphi(T) = k + 2$. □

The sum of the sizes of the sets of the overlap representation produced by the construction in the proof is quadratic. This representation can be efficiently constructed in the sum of the sizes of the sets of the representation, which is $O(nk)$. This yields an algorithm that may not be asymptotically optimal, as a lower bound stronger than $\Omega(n)$ on the sum of the sizes of the sets of a representation for a caterpillar is not known.

3.2.5 Connected Components

In this section we examine the size of a minimum overlap representation for a disconnected graph based on the sizes of minimum overlap representation of each of the connected components of the graph. For general components, this does not produce any new algorithms, but it does allow us to find the minimum overlap representation of a graph composed of the pieces we have already studied. The results here can allow for a divide and conquer approach when trying to find the size of an overlap representation for graphs such as threshold graphs and cographs that can be defined in terms of decomposition schemes. No such algorithms are known, but it is hoped that these results can eventually lead to algorithms for these problems.

To show the desired result, we will make use of a lemma that is an immediate consequence of Lemma 3.13, but is more focused on the present goal. This lemma will be essential to the proof of the theorem that follows, and we will also use it as to demonstrate the hardness of a problem related to finding the overlap number of a graph, which will be done in Section 3.3.3.

Lemma 3.19. *If B_1, B_2 are two connected components of a graph G such that $|B_1|, |B_2| \geq 2$, then in any overlap representation $\mathcal{C} = \{S_v : v \in V\}$ of G, for some $i \in \{1,2\}$ no set S_v for $v \in B_i$ contains any set S_w for $w \in B_{3-i}$. Furthermore, where $U = \bigcup_{v \in B_i} S_v$, any set S_w with $w \in B_{3-i}$ either contains U, or is disjoint from it.*

Proof. If for any $x \in B_1$ and $y \in B_2$ the sets S_x and S_y are disjoint, then there is nothing to prove. We will assume, without loss of generality, that there is $x \in B_1$ and $y \in B_2$ such that $S_x \subseteq S_y$. Setting $X = B_1$ and $Y = B_2$, a direct application of Lemma 3.13 yields the desired result. □

With this lemma, we are able to prove a theorem relating the overlap number of a graph to the overlap numbers of the components of the graph.

Theorem 3.20. *If G is a graph with connected components $B_1, B_2, \ldots, B_k$*

$$\varphi(G) = \sum_{i=1}^{k} \varphi(B_i) - (k-1).$$

Proof. If $k = 1$, the theorem is trivially true. We will assume that all components of $G = (V, E)$ have size at least two, as isolated vertices can be added to a nonempty graph without increasing the size of the overlap representation, by simply assigning the isolated vertex a set consisting of any single element. In the case that G consists only of isolated vertices, the theorem is also trivially true. To prove this theorem we will first, as before, show a lower bound, and then argue that a representation achieving this lower bound must exist.

In the case that $k = 2$, we can use Lemma 3.19 to immediately obtain a lower bound. By the lemma, the two components must either be independent, with no elements in common in the overlap representation, or some sets of one component can contain the sets of the other. If the two components are independent then the size of a minimum overlap representation is clearly given by $\varphi(G) \geq \varphi(B_1) + \varphi(B_2)$. In the other case, we assume that not only the sets associated with vertices of B_2 may contain sets associated with vertices of B_1, but that in doing so, by Lemma 3.19, any set associated with a vertex of B_2 that intersects the set U of elements in the union of the sets associated with the vertices of B_1, must contain the all of U. In this case the elements of U act as a single element, without any further restrictions, and so given a minimum overlap representation for G, we can take the representation restricted to B_2 and replace the elements of U by a single new element, resulting in an overlap representation for G of size $\varphi(B_2) \leq \varphi(G) - \varphi(B_1) + 1$, which yields the desired bound of

$$\varphi(G) \geq \varphi(B_1) + \varphi(B_2) - 1.$$

In the case that $k \geq 3$, we consider a minimum overlap representation $\mathcal{C} = \{S_v : v \in V\}$, and we will once again show a lower bound on the size of $\mathcal{C}$. Take any three components, and, for clarity, let the vertex sets of these components be A, B, and C. If some set associated with a vertex of A is contained in a set, S_b for $b \in B$, and some set associated with not necessarily the same vertex of A is contained in S_c for $c \in C$, then, by Lemma 3.19 the sets S_b and S_c must contain $\bigcup_{a \in A} S_a$. In particular, S_b and S_c intersect, and so one set must contain the other, as they are sets associated with nonadjacent vertices in G. This forces a containment relationship between B and C, so that the set associated with any vertex of A is forced to be contained in the sets associated with the vertices of one of B or C by transitivity. To see how this observation is useful, we build a graph F, where the vertices of the graph are components in G, and two vertices A and B are connected by a directed edge if there is some vertex $a \in A$ and $b \in B$ such that $S_a \subseteq S_b$ in $\mathcal{C}$. Notice that by Lemma 3.19, each pair of vertices is either nonadjacent, or connected by one directed edge. The above observation is then simply the observation that no vertex, v, of F is connected to two nonadjacent vertices by edges directed away from v. This implies that if we take the transitive reduction of F', we obtain a graph with no cycles, and this graph remains acyclic even if we discard the orientation of the edges. Let F be the directed forest resulting from this transitive reduction. Since the edges of F represent containment and no vertex is connected by directed edges to two nonadjacent vertices, each tree has a unique root that all edges of the tree are directed towards.

As in the case that $k = 2$, if two components B_i and B_j are related by containment such that $S_i \subseteq S_j$ for some $i \in B_i, j \in B_j$, the elements of $U = \bigcup_{v \in B_i} S_v$ function as a single element in the representation for the vertices of B_j, which is otherwise unrestricted. Thus if we take an overlap representation of these two components we are able to find a representation that is at most one element smaller than the representation of the two components by disjoint sets. Notice that we can save this one element once for every edge of F, as these edges count exactly the containment relationships that are not forced by transitivity. The largest number of edges F can have is one fewer than the number of components of G, as there must be some root vertex that is not connected by a directed edge to any other vertex. This provides the following lower bound,

$$\varphi(G) \geq \sum_{i=1}^{k} \varphi(B_i) - (k-1). \tag{3.6}$$

To show that a representation exists that achieves this bounds, we take a minimum overlap representation for each component B_i of G, such that any two of these representations are disjoint. We then, for each i in increasing order, create a containment relationship between B_i and B_{i+1}, by choosing an arbitrary element of the representation for B_{i+1} and replacing it with the union of all elements used in the representation of B_i. The resulting representation is a valid overlap representation for G, as we have replaced elements in such a way as to not affect the overlapping properties within a component, and, given any two components, if two sets of the representations associated with them have nonempty intersection, then one set must contain the other, so that there are no adjacencies created between components. Notice that this representation has size given by Equation (3.6), as we have taken optimal representations for each component, and removed exactly $k - 1$ elements, and so this is an optimal overlap representation for G, of size $\sum_{i=1}^{k} \varphi(B_i) - (k-1)$, which proves the theorem. □

From this theorem, we extract the following corollary, which considers the case of the disjoint union of two graphs. The proof of this corollary is a direct application of the theorem.

Corollary 3.21. *If G and H are graphs, and $G + H$ is the disjoint union of G and H, then*

$$\varphi(G+H) = \varphi(G) + \varphi(H) - 1.$$

Proof. Let $A_1, A_2, \ldots, A_{k_1}$ be the components of G and let $B_1, B_2, \ldots, B_{k_2}$ be the components of H. We have, by Theorem 3.20, the following two equations

$$\begin{aligned} \varphi(G) &= \sum_{i=1}^{k_1} \varphi(A_i) - (k_1 - 1), && (3.7) \\ \varphi(H) &= \sum_{i=1}^{k_2} \varphi(B_i) - (k_2 - 1). && (3.8) \end{aligned}$$

Also by this theorem, since the components of $G + H$ are exactly the components of G and H, we know that

$$\varphi(G+H) = \sum_{i=1}^{k_1} \varphi(A_i) + \sum_{i=1}^{k_2} \varphi(B_i) - (k_1 + k_2 - 1),$$

and by combining this with equations (3.7) and (3.8), we obtain

$$\varphi(G+H) = \varphi(G) + \varphi(H) - 1,$$

as desired. □

3.3 Hardness Results

In this section we will examine some **NP**-completeness results for problems related to finding the minimum overlap representation of a given graph. This problem is not known to be hard, but given the hardness of these related problems, and the relationship between overlap and intersection representations, for which this problem is known to be hard, it would be unusual for the problem to admit a polynomial-time algorithm. For completeness, a formalization of this problem is given below.

Problem. The OVERLAP NUMBER problem is defined as:

Instance: A graph, $G = (V, E)$, and an integer k.

Question: Is there an overlap representation $\mathcal{C} = \{S_v : v \in V\}$ of G with

$$\left| \bigcup_{v \in V} S_v \right| \leq k?$$

In Section 3.3.1 we will see that the problem of extending an overlap representation is **NP**-complete, in contrast to the polynomial-time algorithm for the related problem on intersection representations, which was presented in Section 2.3.5. In Section 3.3.2, we will show the hardness of the problem of finding a minimum overlap representation with no containment between the sets of the representation, by relating this problem to the problem of finding a minimum intersection representation. We will then conclude this section with Section 3.3.3, where we will show that this problem remains hard if we allow a constant number of containment relationships in the representation.

3.3.1 Extending an Overlap Representation

A natural approach to finding the overlap number for a graph is to employ a greedy strategy, adding one vertex at a time, and only making changes to the set associated with the newly added vertex. Unfortunately, this is not a feasible approach for a general graph, as the problem of deciding whether or not a new element needs to be added to the representation is **NP**-complete, even in the case that all sets of the representation share a common element. A formalization of the problem of deciding if the new vertex can be represented without adding any elements, or altering the given representation of the other vertices, is given below.

Problem. The OVERLAP EXTENSION problem is defined as:

Instance: A graph, $G = (V, E)$, an overlap representation $C = \{S_v : v \in V\}$ of G, and a set $A \subseteq V$.

Question: Is there a set $S \subseteq \bigcup_{v \in V} S_v$ that overlaps S_v if and only if $v \in A$?

The STAR OVERLAP EXTENSION problem is defined in the same way, except we require that all sets in $C \cup S$ have an element in common. Both of these problems are **NP**-complete, even in the case that $A = V$, as the reductions that follow will show. Before presenting the first of these reductions, we must introduce the problem that we will reduce an instance of OVERLAP EXTENSION to. This is the NOT-ALL-EQUAL 3SAT, which is similar to the usual 3SAT problem, except that we seek a satisfying truth assignment where no clause has all true literals. This problem is defined in Appendix A, and is known to be **NP**-complete [49].

Using the NOT-ALL-EQUAL 3SAT problem, we can show the **NP**-hardness of the OVERLAP EXTENSION problem by demonstrating a polynomial time transformation to instances of the overlap problem from the satisfiability problem that produces a positive instance of the satisfiability problem if and only if the instance of the overlap problem is also solvable. This is done in the proof of the following theorem.

Theorem 3.22. OVERLAP EXTENSION *is* **NP**-*complete.*

Proof. Given an instance $(G, \mathcal{C}, A)$, and a set S, we can easily check, in polynomial time, if S satisfies the required conditions, and so the problem is clearly in **NP**.

For the reduction, we will transform an instance of NOT-ALL-EQUAL 3SAT into an instance of OVERLAP EXTENSION. Let (U, F) be the instance of NOT-ALL-EQUAL 3SAT, where $U = \{x_1, x_2, \dots, x_n\}$ is the set of variables, and $F = \{c_1, c_2, \dots, c_m\}$ is the set of clauses, where we have, for each i, $|c_i| = 3$. In the case that $n < 4$, we examine all possible truth assignments to determine if there is a solution to the NOT-ALL-EQUAL 3SAT instance. If there is a solution, we output the instance where $G = (V, E)$ is a path of length two, and $\mathcal{C} = \{\{0,1\},\{1,2\}\}$, with $A = V$, as this instance has solution $S = \{0, 2\}$. If, on the other hand, there is no solution to the satisfiability problem, we output the instance where $G = (V, E)$ consists of a single vertex, $\mathcal{C} = \{\{0\}\}$, and $A = V$, as there is no set S that overlaps $\{0\}$, so this is also a no instance of OVERLAP EXTENSION.

In the case that $n \geq 4$, we will construct a graph $G = (V, E)$, an overlap representation $\mathcal{C}$ of G, and a set A, to form an instance of OVERLAP EXTENSION. The vertices in the graph we construct are given by

$$V = \{v_i : 1 \leq i \leq n\} \cup \{w_i : 1 \leq i \leq m\},$$

where each vertex v_i will be associated with a variable $x_i \in U$, and each vertex w_i will be associated with a clause $c_i \in F$. Having vertices, we can then introduce the overlap representation, $\mathcal{C}$, in the instance of OVERLAP EXTENSION, which is given by

$$\mathcal{C} = \{S_{v_i} = \{x_i, \neg x_i\} : x_i \in U\} \cup \{S_{w_i} = c_i : c_i \in F\}.$$

The sets associated with vertices w_i are simply the clauses of the satisfiability problem, which are sets of three literals, and the sets associated with vertices v_i contain both literals connected to the associated variables in U.

Having defined the vertices and the overlap representation, we simply set the edge set to the set of edges that agrees with the overlap representation $\mathcal{C}$, which is given by $E = \{(u, v) : S_u \text{ and } S_v \text{ overlap}\}$. Finally, we take $A = V$ and $G = (V, E)$, to form an instance $(G, \mathcal{C}, A)$ of OVERLAP EXTENSION. This transformation can clearly be computed in polynomial time. Notice that this instance is a yes instance if and only if there is some set S of literals in the satisfiability problem such that S overlaps S_v for all $v \in V$. We will show that such any such S forms a satisfying truth assignment for (U, F) in which each clause has at least one false literal.

To see this, let $S \subseteq U \cup \{\neg x : x \in U\}$ be a set that overlaps all elements of $\mathcal{C}$. Since S overlaps each $S_{v_i} = \{x_i, \neg x_i\}$, S must contain exactly one element of S_{v_i}. This forces S to choose a truth value for each variable x_i, as we can consider the truth assignment that sets every literal in S to true. In addition, we know that S overlaps each $S_{w_i} = c_i$, which forces at least one, but not all, of the literals in c_i to be contained in S. Viewing S as a truth assignment, it must make true at least one, but not all, literals of the clause c_i, which proves that if the instance of OVERLAP EXTENSION is a yes instance, then the initial instance of NOT-ALL-EQUAL 3SAT is also a yes instance.

In the other direction, we take any truth assignment T that satisfies the original NOT-ALL-EQUAL 3SAT instance without making all literals in any clause true, and set S to be the set of all literals made true by T. Since T is a truth assignment, for each $1 \leq i \leq n$, S contains exactly one of x_i and $\neg x_i$, and so S overlaps S_{v_i} for all i. Furthermore, since T is a satisfying truth assignment, S must intersect each S_{w_i}. In addition, S cannot contain any S_{w_i}, as this would imply that T satisfies all literals of clause c_i. Finally, $|S_{w_i}| = |c_i| = 3$, and $|S| = |U| \geq 4$, so S_{w_i} cannot contain S for any i. This implies that S overlaps S_{w_i} for all $1 \leq i \leq m$, and so S is a solution to the instance of OVERLAP EXTENSION. We have shown that there is a solution to the original instance of NOT-ALL-EQUAL 3SAT if and only if there is a solution to the constructed instance of OVERLAP EXTENSION, and so we conclude that, since the first problem is **NP**-hard, the OVERLAP EXTENSION problem is **NP**-complete. □

In the case of an overlap representation where there is an element common to each set, we can prove a similar theorem. In this case we will present a reduction to this problem from the well-known 3SAT problem. This problem is defined in Appendix A, and is also known to be **NP**-complete [32]. Using the 3SAT problem, we can show that the STAR OVERLAP EXTENSION problem is **NP**-complete, which is the content of the following theorem.

Theorem 3.23. STAR OVERLAP EXTENSION *is* **NP**-*complete.*

Proof. Once again, this problem is clearly in **NP**, as we can, given a set S, verify that it overlaps with the desired sets of the given overlap representation, and check that there is an element in S that is common to each set of the representation. To show the hardness of this problem, we will use a reduction similar to the one used in the proof of Theorem 3.22. We will transform an instance, (U, F), of 3SAT, where U is the set of variables and F is the set of clauses, to an instance $(G, \mathcal{C}, A)$ of STAR OVERLAP EXTENSION. Once again we separate the case where $|U| = n < 4$, by testing all possible truth assignments to the variables to determine the satisfiability of the given 3SAT instance. If the instance is satisfiable, we output the instance of STAR OVERLAP EXTENSION given by two connected vertices, with overlap representation given by $\{\{0,1,2\},\{0,1,3\}\}$, where A consists of both of the vertices of the graph. The set $S = \{0,2,3\}$ then satisfies the requirements of the STAR OVERLAP EXTENSION problem. If the given instance of 3SAT is not satisfiable, we output the instance given by a single vertex, with associated set in the representation $\{0\}$, where the set A contains this vertex. As no set will overlap $\{0\}$, this instance is a no instance of STAR OVERLAP EXTENSION, as desired.

Given an instance (U, F) of 3SAT, where the set $U = \{x_1, x_2, \ldots, x_n\}$ for $n \geq 4$, and $F = \{c_1, c_2, \ldots, c_m\}$ such that $|c_i| = 3$ for all i, we construct an equivalent instance of STAR OVERLAP EXTENSION as follows. We will use the same vertex set as in the proof of Theorem 3.22, so that

$$V = \{v_1, v_2, \ldots, v_n\} \cup \{w_1, w_2, \ldots, w_m\},$$

where the vertex v_i will be associated with the variable x_i and the vertex w_i with the clause c_i. We also form the overlap representation $\mathcal{C}$ in a similar way, where we let $L = U \cup \{\neg x_i : x_i \in U\}$ be the set of all possible literals formed from the variables in U, and we set

$$\begin{aligned} S_{v_i} &= \{0, x_i, \neg x_i\} \\ S_{w_i} &= \{0\} \cup (L \setminus c_i), \end{aligned}$$

where, once again, $\mathcal{C} = \{S_{v_i} : 1 \leq i \leq n\} \cup \{S_{w_i} : 1 \leq i \leq m\}$. As before, we let E be the edge set determined by this overlap representation, setting $E = \{(u, v) : S_u \text{ and } S_v \text{ overlap}\}$. Finally, we set $G = (V, E)$ and $A = V$, so that we have an instance $(G, \mathcal{C}, A)$ of STAR OVERLAP EXTENSION. This transformation is clearly polynomial-time computable.

To see that (U, F) is satisfiable if S is a set that both overlaps and shares a common element with all sets of $\mathcal{C}$, we let S be such a set. Since $\{0\} = S_{v_i} \cap S_{v_j}$ for any $i \neq j$, we must have $0 \in S$ as the element in common to all sets of $\mathcal{C}$. Then, since S overlaps $S_{v_i} = \{0, x_i, \neg x_i\}$, and $0 \in S$, we cannot have both of x_i and $\neg x_i$ in S. This allows us to once again form a truth assignment that makes true all literals in S. This truth assignment is, in general, not complete, but we can extend it to all the variables in U by making arbitrary choices for those variables not in S. Since S overlaps each $S_{w_i} = \{0\} \cup (L \setminus c_i)$, there must be some element in S that is not in $L \setminus c_i$, and so we have $S \cap c_i \neq \emptyset$ for all clauses c_i. This is exactly the requirement that the truth assignment defined by S satisfies all clauses of the 3SAT instance, as desired.

Given a satisfying truth assignment for (U, F), the instance of 3SAT, we can find a set S that satisfies the requirements of the STAR OVERLAP EXTENSION problem. We do this by taking S to be the set of literals made true by the satisfying assignment, plus the element 0 to ensure that there is a common element in each set of the overlap representation. Since a truth assignment cannot make both x_i and $\neg x_i$ true, S cannot contain any set $\{0, x_i, \neg x_i\} = S_{v_i}$, and S cannot be contained in such a set as there are at least four variables in the 3SAT instance. Thus S overlaps S_{v_i} for all i. Also, S cannot be contained in $S_{w_i} = \{0\} \cup (L \setminus c_i)$, since if it was, the truth assignment would not satisfy the clause c_i, and S_{w_i} cannot be contained in S, as there is some i such that x_i and $\neg x_i$ belong to S_{w_i}, since $|c_i| = 3$, and there are at least four variables in U. Thus S must also overlap S_{w_i} for all i. This proves that the instance (U, F) of 3SAT is satisfiable if and only if (G, C, A) is a yes instance of STAR OVERLAP EXTENSION, and so, since 3SAT is **NP**-hard, we have shown that STAR OVERLAP EXTENSION is **NP**-complete. □

3.3.2 Containment-Free Representations

In this section we show the hardness of the problem of finding a minimum containment-free overlap representation. This problem is formalized as follows:

Problem. The CF-OVERLAP NUMBER problem is defined as:

Instance: A graph, $G = (V, E)$, and a natural number k.

Question: Does the graph G have a containment-free overlap representation of size k?

In order to show the hardness of finding a minimum containment-free overlap representation, we can make use of the observation that the containment-free overlap representations are exactly the containment-free intersection representations. This is because, in the absence of containment, the definitions of set overlap and intersection coincide. We can then apply a simple reduction from the problem of finding a minimum intersection representation, given as INTERSECTION NUMBER in Section 2.2, to show the **NP**-hardness of the problem of finding a minimum containment-free overlap representation.

Theorem 3.24. CF-OVERLAP NUMBER *is* **NP**-*complete.*

Proof. We will show the **NP**-hardness of this problem by reducing the problem INTERSECTION NUMBER to it, which is shown to be **NP**-hard in [34], as we saw in Section 2.2. Given an instance $G = (V, E)$ and k of INTERSECTION REPRESENTATION, where $n = |V|$, we construct the graph G' by adding, for each $v \in V$, a vertex v' that is only connected to v. More formally, we construct $G' = (V \cup V', E \cup E')$ where

$$\begin{aligned} V' &= \{v' : v \in V\} \\ E' &= \{(v, v') : v \in V\}. \end{aligned}$$

The instance of CF-OVERLAP NUMBER is then given by G' and $k + 2n$.

Notice that, by the comments preceding the theorem, any containment-free overlap representation forms a containment free intersection representation. Notice further that in any containment-free intersection representation for G, the sets S_v and $S_{v'}$ associated with a vertex $v \in V$ and $v' \in V'$ must share a common element, as these vertices are connected, and furthermore, since v' is not connected to any other vertex, this element will only be found in S_v and $S_{v'}$. The set $S_{v'}$ must also contain at least one other element, to prevent it from being contained in S_v, and notice once again by the fact that v' is adjacent only to v, this element is unique to $S_{v'}$. Then, since for each $v \in V$, these two elements are only contained in S_v and $S_{v'}$, we ensure that any intersection representation of G' is containment-free. This leaves the remaining elements of the representation to form an intersection representation for G. Furthermore, since any intersection representation will do, a minimum containment free representation for G' will have size given by $\theta_e(G) + 2n$, where $\theta_e(G)$ is the size of a minimum intersection representation for G. Thus G has an intersection representation of size k if and only if G' has a containment-free intersection representation of size $k + 2n$.

This transformation can clearly be performed in polynomial time, and, given a graph $G = (V, E)$ and a collection $C = \{S_v : v \in V\}$, it is easy to check in polynomial time whether C forms a containment-free overlap representation for G. $\square$

3.3.3 Overlap Representations with Limited Containment

In this section, we consider the problem of finding, for a given graph, a minimum overlap representation with not more than a given number of containment relationships between sets of the representation. Without the constant bound on the number of containment relationships that we impose, this would be a more general problem than that of finding a minimum overlap representation, in which the only bound on the number of containments is given by the number of non-edges of the graph. Formalized as a decision problem, we consider the following problem. The factor of $1/2$ appears since the non-edges (u,v) and (v,u) are both counted, and so to count containment relationships we halve the number of pairs of non-edges with intersecting sets.

Problem. The L-CONTAINMENT OVERLAP REPRESENTATION problem, for any natural number L, is defined as:

Instance: A graph, $G = (V, E)$, and a natural number k.

Question: Is there is some collection $\mathcal{C} = \{S_v : v \in V\}$ that forms an overlap representation, satisfying $\left|\bigcup_{v\in V} S_v\right| \leq k$ and

$$\frac{1}{2}\left|\{(u,v) \notin E : u \neq v \text{ and } S_u \cap S_v \neq \emptyset\}\right| \leq L?$$

These restrictions are merely that the number of elements in the representation must be at most k, and there can be at most L non-edges represented in the overlap representation by containment. This problem, when $L = 0$ is exactly the CF-OVERLAP NUMBER problem, and so it is **NP**-complete in this case, by Theorem 3.24. We can also show that this problem is **NP**-complete for any value of L. A simple Turing reduction from the CF-OVERLAP NUMBER problem is given by making $2L+1$ copies of the input graph, and then finding an overlap representation with no more than L containments, which, by the pigeonhole principle, must leave at least one copy of G containment free, both internally, and with respect to other components of the graph. Furthermore, if we have a minimum representation, then this representation for G must also be minimum, as the sets associated with the vertices of this copy of G are disjoint from the sets associated with vertices in any other copy. We can then output the size of the representation restricted to this copy of G, which is a minimum containment-free overlap representation of G. With a little more work, we can find a many-one reduction from the CF-OVERLAP NUMBER problem, by adding to the graph G extra components where a minimum representation will be compelled to "spend" all L set containments, leaving G with a containment-free representation. If we are careful, we can do this in such a way that we can track the number of elements these extra components will add to the representation, so that we find the size of a minimum containment-free overlap representation of G. To show this, we will make use of Lemma 3.14, which will allow us to show an upper bound on the number of elements we will be able to save by allowing containment relationships between components of the constructed graph.

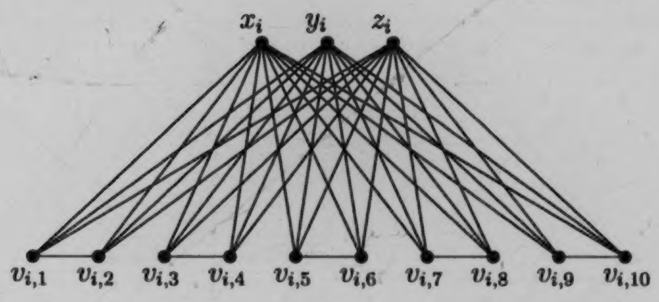

Figure 3.2: Example of B_i with $n = 4$.

Theorem 3.25. *For any* $L \in \mathbb{N}$, L-CONTAINMENT OVERLAP REPRESENTATION *is* **NP**-*complete.*

Proof. As mentioned above, we will reduce the CF-OVERLAP REPRESENTATION problem to the L-CONTAINMENT OVERLAP REPRESENTATION problem. To this end, we let $G = (V, E)$ and k be an instance of the containment-free overlap representation problem. We set $n = |V|$, and we consider only cases where $n \geq 4$, as smaller cases can be solved as part of the transformation by searching all possible representations and producing as output a trivial yes or no instance. In the instance we construct, we add $2L$ components to the graph G. Each of these components is given by the graph $B_i = (V_i, E_i)$, which is constructed from $n+1$ disjoint edges, with three nonadjacent universal vertices, as shown in Figure 3.2. More formally, the vertices of each component are

$$V_i = \{v_{i,j} : 1 \leq j \leq 2n+2\} \cup \{x_i, y_i, z_i\},$$

and the edges are given by

$$\begin{aligned} E_i &= \{(v_{i,2j-1}, v_{i,2j}) : 1 \leq j \leq n+1\} \cup \\ &\quad \{(x_i, v_{i,j}), (y_i, v_{i,j}), (z_i, v_{i,j}) : 1 \leq j \leq 2n+2\}. \end{aligned}$$

The graph in the constructed instance of L-CONTAINMENT OVERLAP REPRESENTATION is then given by a disjoint union, $H = G + B_1 + B_2 + \cdots + B_{2L}$, of $2L$ of these new components with the graph given as part of the CF-OVERLAP NUMBER instance. The value k' that also makes up the constructed instance is set to

$$k' = k + 3L(n+1) + 4L(n+1), \tag{3.9}$$

to complete the instance of L-CONTAINMENT OVERLAP REPRESENTATION. This instance can clearly be constructed in polynomial time.

Before showing that the given instance of the CF-OVERLAP REPRESENTATION problem is equivalent to the constructed instance, we first make some observations about overlap representations of the graphs B_i. In any containment-free overlap representation for the vertices $v_{i,j}$ of B_i, we must use at least $3(n+1)$ elements, as each edge $(v_{i,2j-1}, v_{i,2j})$ requires at least three elements, and all of these edges are disjoint, so the sets associated with these edges cannot share any elements. Thus, any minimum containment-free representation uses exactly three elements for each edge $(v_{i,2j-1}, v_{i,2j})$. By the structure of an overlap representation, these three elements are arranged into an element unique to the set associated with $S_{v_{i,2j-1}}$, an element unique to $S_{v_{i,2j}}$, and an element in the intersection of these two sets. We can then extend such a minimum representation to the vertices x_i and y_i without increasing the size of the representation. To do this, we set S_{x_i} to be the collection of all elements in the intersection of the sets associated with the edge, that is

$$S_{x_i} = \bigcup_{1 \leq j \leq n+1} S_{v_{i,2j-1}} \cap S_{v_{i,2j}},$$

and, having set S_{x_i} to the elements common to the representation of each edge, we can set S_{y_i} to the elements that are unique to each vertex, obtaining

$$S_{y_i} = \bigcup_{1 \leq j \leq 2n+2} S_{v_{i,j}} \setminus S_{x_i}.$$

By construction, these two sets must be disjoint, and also, each of these sets must intersect the set associated with each $v_{i,j}$. Furthermore, since $n+1 \geq 2$, these sets will not be contained in any of the sets associated with the vertices $v_{i,j}$. Thus, without increasing the size of the representation, we can extend a minimum containment-free overlap representation for the $v_{i,j}$ to also include the vertices x_i and y_i. In the case that we allow containment between S_{x_i} and S_{z_i}, we can then set $S_{z_i} = S_{x_i}$ to form a minimum overlap representation for B_i, of size $3(n+1)$, subject to the restriction that the non-edge (x_i, z_i) is the only non-edge for which containment is allowed in the representation.

If we seek a containment-free overlap representation for B_i the situation is more bleak, as we still must use three unique elements to represent each edge $(v_{i,2j-1}, v_{i,2j})$, but with only three elements representing an edge, one unique to each endpoint, and one common element, the only three nonempty disjoint subsets we can choose to represent x_i, y_i and z_i are single elements, but two of these sets will not intersect both endpoints of each the edges we consider. Thus we will need to use at least four elements for each of the $n+1$ edges of the form $(v_{i,2j-1}, v_{i,2j})$. This number also suffices, for if we introduce a new element a_j to each set $S_{v_{i,2j-1}}, S_{v_{i,2j}}$, which doubles the number of common elements in each of these sets, we can use the same representation as before for x_i and y_i, and we can now set $S_{z_i} = \{a_j : 1 \leq j \leq n+1\}$ to form a containment-free overlap representation for B_i. Thus, if we are allowed only one containment, we can reduce the number of elements needed to represent B_i from $4(n+1)$ to $3(n+1)$, which will be the key to the proof that the constructed instance of L-CONTAINMENT OVERLAP REPRESENTATION is equivalent to the given instance of CF-OVERLAP NUMBER.

If C is a minimum L-overlap representation for H, of size no more than $k' = k+3L(n+1)+4L(n+1)$, we will show that G has a containment-free overlap representation of size not more than k. We claim that, as C is minimum, the representation C when restricted to G is already containment-free, and to show this, we will examine the potential cases for a non-edge to be represented by containment.

The first such case we consider is any containment within the representation of G, which is, two vertices u and v such that $S_u \subseteq S_v$. We will replace S_v with $n-1$ new elements, $a_1, a_2, \ldots, a_{n-1}$, to obtain $S'_v = \{a_1, a_2, \ldots, a_{n-1}\}$. This removes the containment relationship between u and v, and forces S'_v not to contain any other set in the representation. In order to ensure that the representation still forms a valid overlap representation for H, we will need to modify the sets associated with some of the other vertices. There are exactly three ways a set can interact with S'_v: we can have the set we consider contain S'_v, the two sets can be disjoint, or the two sets can overlap. We cannot have the case that any other nonempty set is contained in S'_v, as it contains only elements new to the representation. We will consider, for each of these three interactions, how to alter the set to maintain a valid overlap representation of H. For any vertex w with $S_v \subseteq S_w$, we replace the set S_w with the set $S'_w = S_w \cup S'_v$, to ensure that this containment relationship is not altered. This alteration does not affect the overlap, containment, or disjointedness relations of the set S_v, as these are new elements, and by transitivity, we have added these new elements to any set that contains S_w. If w is a vertex such that S_w and S_v are disjoint, then S_w and S'_v must also be disjoint, and there is nothing to do in this case. If w is such that S_w and S_v overlap, then we have $S'_v \cap S_w = \emptyset$, which we correct by setting $S'_w = S_w \cup \{a_i\}$, for an element $a_i \in S'_v$ that we have not already used for this purpose. This will force S'_w and S'_v to overlap, as the conditions that $n \geq 4$ and S_w overlaps the set S_v ensure that there are least two elements in each of these sets. We must also add the element a_i to any set that contains S_w, to preserve this containment relationship. This will not affect the representation of any vertex but v, as the sets that the element a_i is being added to must also intersect S_v. One potential problem is that we may add all of the a_i to a set that should not contain S'_v, but since there are at most $n-2$ vertices that are adjacent to v, we can have only this many sets overlapping S_v, and so the element a_{n-1} will be present only in the set S'_v, and those sets that also contain S_v. This also ensures that we will not run out of elements $a_i \in S'_v$ when making this change to S_w. Thus, we can remove at least one containment relationship from G, by adding $n-1$ new elements to the representation. Since there are $2L$ components B_i, and only $L-1$ remaining containments, there must be some i for which the vertices of B_i are involved in no containment relationships. We can use the containment we just removed from G to reduce the size of the representation for B_i from $4(n+1)$ to $3(n+1)$, which, in total, will save $n+1-(n-1)=2$ elements from the representation, contradicting the assumption that $\mathcal{C}$ was minimal. Thus, the vertices of G are not involved in any containment relationships in $\mathcal{C}$.

The second case of a containment relationship is one internal to one of the components B_i. If this containment is between two vertices $v_{i,j}$ and $v_{i,k}$, we can simply replace the representation for $v_{i,j}$ and the vertex $v_{i,j+1}$ it forms an edge with (we assume, without loss of generality, by relabelling if necessary, that $v_{i,j}$ and $v_{i,j+1}$ are adjacent). This is done by setting $S_{v_{i,j}} = \{a_1, a_3, a_4\}$ and $S_{v_{i,j}} = \{a_2, a_3, a_4\}$, where the elements a_i are new to the representation. Finally, we add a_1 and a_2 to S_{x_i}, a_3 to S_{y_i}, and a_4 to S_{z_i}, being careful to add these elements to any set that contains these elements (which will preserve the representation of these other elements, since set containment forms a partial order). If preserving these containment relationships results in all of $\{a_1, a_2, a_3, a_4\}$ being contained in one of S_{x_i}, S_{y_1}, or S_{z_i}, which will happen if between each pair of x_i, y_i, and z_i there is a containment relationship, then we simply add a_3 to each of S_{x_i}, S_{y_i} and S_{z_i}, and remove a_1, a_2, and a_4 from these sets, once again being careful to preserve any containment relationships. This replacement removes the containment between $v_{i,j}$ and $v_{i,k}$, and leaves a valid overlap representation. As the cost of this alteration was only four elements, and we can apply the freed containment relationship to some other component B_j to save $n+1$ containments, this case also contradicts the optimality of $\mathcal{C}$.

The only remaining case for a containment internal to B_i is one between two of x_i, y_i, and z_i, as these vertices are adjacent to all other vertices of B_i. We must also consider the case that between the vertices x_i, y_i and z_i there are two or more containments, but since given any containment-free representation of the remaining vertices of B_i, we can find an overlap representation for all of B_i using no new elements and only one containment, this also contradicts the optimality of $\mathcal{C}$, as each extra containment can be used in some other component to save $n+1$ elements.

The final case we must consider is a containment relationship between two vertices in differing components of H. Let U and W be the vertex sets of the two components, where for some vertex $u \in U$ and $w \in W$ we have $S_u \subseteq S_w$. By Lemma 3.13 the set S_w contains all sets associated with vertices of U, and furthermore no set associated with a vertex of U can contain any set associated with any vertex of W. Let $A = \bigcup_{u \in U} S_u$. This lemma implies that for any vertex in $v \in W$, either S_v contains A or it is disjoint from it. The elements of A then, within W, act as a single element. This allows these elements to be replaced with a single new element, where once again, whenever we add a new element to a set we must be careful to also add this new element to any sets that contained the original set. After this replacement has been made, we have removed at least one containment relationship, at a cost of one new element in the representation. In the resulting representation, we have used at most $L - 1$ containment relationships, and there are $2L$ components B_i, so once again, by the pigeonhole principle, there is some component to which we can apply the containment relationship to save $n + 1$ elements, and once again, this contradicts the optimality of C.

Thus, a minimum L-containment overlap representation for H uses containment only between the vertices x_i, y_i, and z_i, and uses at most one containment per triple of vertices. Thus, in a minimum overlap representation, we have a containment-free overlap representation for G, and L of the components given by B_i, and we have an overlap representation using only one containment for the remaining L of the B_i. By the preceding argument, where we let r be the containment-free overlap number of G, this representation has size $r + 4L(n + 1) + 3L(n + 1)$, which by Equation (3.9) is less than k' only when $r \leq k$, as desired.

Fortunately, the other direction is simple. If we take any containment-free overlap representation for G of size no more than k, we can form the representations discussed above for each B_i, by simply using three elements per edge $(v_{i,2j-1}, v_{i,2j})$ for L of the B_i and four elements per edge for the remaining L. Placing the containments in appropriate places, we can find an L-containment overlap representation for H of size no more than $k + 3L(n + 1) + 4L(n + 1) = k'$, as required. □

3.4 Bounds on the Size of an Overlap Representation

In this section we examine upper bounds on the size of any minimum overlap representation. In contrast to the intersection case, here we do not know of any graphs that require a quadratic number of elements, as is required in the intersection case by a theorem of Erdős, Goodman, and Pósa [16], which is given here as Theorem 2.16.

It is interesting to note that while the adding Helly property limits the number of cliques in an intersection graph, if we consider an overlap graph, no such limitation occurs. As an explicit example, we construct a small Helly overlap representation for $O_{n/2}$, the complement of a perfect matching, where a perfect matching is simply the union of $n/2$ disjoint edges, an example of which appears as Figure 2.6. To do this, we will take $2n$ vertices, $\{1, \ldots, 2n\}$, and separate them into two groups. To vertex i, where $1 \leq i \leq n$, we assign the set $\{0, i\}$. To vertex $i + n$, where again $1 \leq i \leq n$, we also assign the set $\{0, i\}$. Then, in the overlap graph, vertices i and $i + n$ are adjacent to every vertex j or $j + n$, for $j \neq i$, since the intersections of the two sets contain the element 0, and j is not included in the set for vertex i or $i + n$. Also, since the sets assigned to the vertices i and $i + n$ are the same, they are not adjacent in the overlap graph. Thus, each vertex in this graph is adjacent to all but one vertex, and these nonadjacent vertices are paired together. This graph is exactly the complement of a perfect matching, or in the notation of Section 2.5, this is exactly the graph $O_{n/2}$, so we know that it contains $2^{n/2}$ maximal cliques, and yet it has overlap number no larger than $n + 1$. While the number of cliques may remain unbounded by a polynomial in terms of the length of the overlap representation, there is an algorithm for finding the maximum clique of an overlap graph with an overlap model with the Helly property, with runtime polynomial in the size of the representation, as shown in [9].

In Section 3.4.1 we will examine the connection between the size of a minimum intersection representation and the size of a minimum overlap representation for any graph in a given class of graphs. This will allow us to find several bounds on the maximum size of a required overlap representation for several classes of graphs. The case of cocomparability graphs, which we will study in Section 3.4.2, will require a different technique. This is an interesting case, as the graph used to prove that some graphs have quadratic intersection number is a cocomparability graph, and as we shall see, all cocomparability graphs have overlap representations using only a linear number of elements.

In fact, the graph with maximum intersection number, as shown in [16], is a complete bipartite graph. This graph does not have quadratic overlap number, however, as by Proposition 3.8, any complete bipartite graph has overlap number given by the overlap number of K_2, which is three.

3.4.1 Relation to the Size of an Intersection Representation

In this section, we examine a simple method to transform any intersection representation to an overlap representation. We will be able to do this by adding only a linear number of elements. While these representations will rarely be minimum overlap representations, they do allow us to make asymptotic bounds on the size of the required overlap representations for many classes of graphs. This simple method is given as the following proposition.

Proposition 3.26. *If a graph G with n vertices has an intersection representation of size k, then the minimum overlap representation of G has size at most $k + n$.*

Proof. If we let $G = (V, E)$ have intersection representation $\mathcal{C} = \{S_v : v \in V\}$, we build the representation $\mathcal{D} = \{S'_v = S_v \cup \{v\} : S_v \in \mathcal{C}\}$, where we assume the elements v that we add are new to the representation.

As $\mathcal{C}$ is an intersection representation, we have $S_u \cap S_v = \emptyset$ if and only if $(u, v) \in E$. However, since we have added a new element to each set S_v to form the set S'_v, S'_u and S'_v are still disjoint if and only if S_u and S_v are. Also, if S_u and S_v intersect, then so must S'_u and S'_v, and in fact, they must overlap, as for any $w, z \in V$ we have $w \in S'_v$ if and only if $z = w$, which eliminates any set containment from the representation. □

Using this proposition, we can immediately apply bounds on the number of maximal cliques from Section 2.5.2, which must bound the maximum size of any minimum intersection representation, to find bounds on the maximum size of any minimum overlap representation in a given class of graphs. The first of these results is a bound on the size of a minimum overlap representation for any chordal graph.

Corollary 3.27. *If G is a chordal graph, then G has a minimum overlap representation of size at most $2n$.*

Proof. By a direct application of Proposition 3.26 to the bound of n on the number of maximal cliques in any chordal graph in Proposition 2.19, we obtain the desired bound of $2n$. □

Another class of graphs for which we can very easily state a linear bound on the overlap number is the planar graphs.

Corollary 3.28. *If G is a planar graph, then G has a minimum overlap representation of size at most $\frac{10}{3}n - 6$.*

Proof. We will once again apply Proposition 3.26, in this case to the bound of Prisner [43], given in Theorem 2.23. This bound limits the number of maximal cliques in any planar graph to $\frac{7}{3}n - 6$, which serves as an upper bound on the size of a minimum intersection representation. The application of Proposition 3.26 to this bound produces the bound of $\frac{10}{3}n - 6$. ☐

More generally, Proposition 3.26 allows us to show the following bound, which we have no evidence to show is asymptotically tight.

Corollary 3.29. *If G is any graph, then G has a minimum overlap representation of size no more than* $\left\lfloor \frac{n^2+4n}{4} \right\rfloor$.

Proof. By the theorem of Erdős, Goodman, and Pósa [16]], given here as Theorem 2.16, we have a bound on the intersection number of any graph of $\lfloor n^2/4 \rfloor$. Applying Proposition 3.26 to this bound gives the bound in the statement of the corollary. ☐

We can also use this technique to bound the overlap number of any tree, as any intersection representation for a tree on n vertices must have size exactly $n - 1$, as this is the number of cliques in any edge clique cover, since each edge is in exactly one maximal clique. Using Proposition 3.26 we obtain a bound of $2n - 1$. This bound is not optimal, however, as we can show the following bound, which is met by the tree on two vertices, but is not necessarily asymptotically optimal. A family of trees that does not require $n + 1$ elements in an overlap representation is given by the caterpillars, which, by Corollary 3.18, never require more than n elements.

Proposition 3.30. *Let T be a tree on n vertices, then the size of a minimum overlap representation for T is at most $n + 1$.*

Proof. Let T be a tree; we will form an overlap representation of size $n + 1$ for T. The set S_v of the representation we will construct will be assigned to the vertex v in the tree. To form this representation, we first choose a root, r, arbitrarily, from which we will explore the tree in depth-first order. We assign to the root the set $S_r = \{1, 2\}$, and we will refer to the element 2 as the "new" element of S. We then perform a depth-first search, assigning to each vertex a set containing a new element, and the element that was new in the set associated with the parent vertex. As an example, we assign to the first child of the root the set $\{2, 3\}$, where 3 is the new element in this set. This will ensure that adjacent nodes in the tree have overlapping sets, and that nodes that do not share the same parent are disjoint.

This does not quite produce an overlap representation of the graph, however, as any two children of the same node will have overlapping sets assigned to them. We can repair this defect with a small modification to the construction of the representation. When we assign a set to a vertex, v, with siblings (nodes with the same parent) $w_1, w_2, \ldots, w_k$, we also include in S_v any element assigned to any set in a subtree rooted at any of the w_i that have already been visited. This removes the overlapping of siblings in the tree, as each successive sibling will contain all elements assigned to the subtree of the previous siblings. The children of the node v are not affected by this, as they will contain only elements that did not appear in any set that has previously been assigned.

This representation thus forms an overlap representation for T, and since there is one "new" element introduced for each vertex, and one other element assigned to the root, we have used exactly $n + 1$ elements, as desired. □

3.4.2 Cocomparability Graphs

The case of cocomparability graphs is interesting, due to the fact that the intersection number of a cocomparability graph can have quadratic size. In contrast to this, we can prove a linear bound on the overlap number of any cocomparability graph.

Theorem 3.31. *If G is any cocomparability graph, then G has overlap number at most $n+1$.*

Proof. Given a cocomparability graph G, we will find an overlap representation of size $n+1$. To do this, we transitively orient the non-edges of G. We then construct a topological ordering of $\overline{G}$, which is a removal ordering of the vertices of G such that each vertex in the ordering is a source in $\overline{G}$ when it is removed (vertex with all adjacent non-edges directed outwards). Such an ordering can be constructed for any cocomparability graph.

We next take the topological ordering and reverse it, to obtain an ordering, $v_1, v_2, \ldots, v_n$ of the vertices of G that allows us to add them to the graph one at a time, so that each newly added vertex is a source, with respect to the transitive orientation of the non-edges of G. We will construct the overlap representation in the order given by the reversed topological ordering in such a way that the transitive orientation of the non-edges is respected by containment relationships between the sets of the representation. To the first vertex we assign the set $S_{v_1} = \{0, 1\}$. For each successive vertex v_i, we assign it the set $S_{v_i} = \{0, i\}$, and for each vertex v_j with $j < i$ that is not adjacent to v_i, we set $S_{v_j} = S_{v_j} \cup \{i\}$.

To see that this is a correct representation, we note first that the representation for the graph containing only v_1 with the set $\{0, 1\}$ is a valid representation, with set containment along all directed non-edges of the graph. Then, we assume by induction that the sets we have assigned to all vertices up to v_{i-1}, for any $i > 1$, forms a correct overlap representation for the graph induced on the vertices $v_1, v_2, \ldots, v_{i-1}$, and furthermore, for any two vertices that are not adjacent, the sets assigned to these vertices have a containment relationship, in agreement with the orientation of the non-edge, that is, if (v_j, v_k) is a directed non-edge, then $S_{v_j} \subseteq S_{v_k}$. To see that the changes to the representation we make when adding v_i leave a valid representation, notice that any vertex v_j not adjacent to v_i must have a directed non-edge of the form (v_i, v_j), and by construction, since all sets share the element 0, we have $S_{v_i} \subseteq S_{v_j}$. Also, for any vertex v_j adjacent to v_i, we have $0 \in S_{v_i} \cap S_{v_j}$, and each set has an element not in the other set, since $j \notin S_{v_i}$ and $i \notin S_{v_j}$, so these sets overlap. Since v_i is a source in the graph, these are all the cases we must consider regarding set interaction with S_{v_i}. The only remaining property to verify is that the overlap representation of the previous $i - 1$ vertices has not been affected. Since we have only added an element that is new to the representation to some of the sets, we cannot have affected any overlap relationships, and by construction there are no disjointedness relationships in the representation. This leaves only containment relationships, which can only be affected if $S_{v_j} \subseteq S_{v_k}$ before adding the element i to S_{v_j} and not S_{v_k}. This, however, is a contradiction, as in this case we have the directed non-edges (v_i, v_j) and (v_j, v_k), but not (v_i, v_k), which violates the requirement that we have a transitive orientation of the non-edges of G. Thus, we have a correct representation after adding vertex v_i to the graph, and so, by induction, we have a valid overlap representation for G after we have added v_n to the graph.

This representation has size given by $n+1$, as we added one new element for each vertex after v_1, and two elements with the first vertex. This proves the required upper bound. □

This bound is tight in at least the case of P_2, a single edge, which has overlap number three. We will see another method to prove this bound in Section 4.4, where we will see both upper bounds on the size of any containment representation, and a method to transfer upper bounds on the size of the containment number of $\overline{G}$ to upper bounds on the size of the overlap number of G.

Chapter 4

Containment Representations

4.1 Introduction

Containment representations are another representation that has not been studied as thoroughly as intersection representations, despite the fact that set containment is a natural means to represent the edges of a graph. In order to provide a formal treatment of containment representations we make use of the following definition of a containment representation of a graph. This is merely a formalization of the notion that vertices in a graph are adjacent if and only if, for the sets of a representation associated with them, one set contains the other.

Definition 4.1. Given a graph $G = (V, E)$, a collection $\mathcal{C} = \{S_v : v \in V\}$ is a *containment representation* for G if for any $u, v \in V$ we have

$$(u, v) \in E \text{ if and only if } S_u \subseteq S_v \text{ or } S_v \subseteq S_u.$$

We define the size of a representation as the number of elements in the union of all sets in the collection, which is

$$\left| \bigcup_{v \in V} S_v \right|,$$

and we let the *containment number*, $\xi(G)$, be the size of a minimum containment representation for the graph G.

Once again, in such a representation we do not require that different vertices be assigned different sets. This allows the same set to represent all vertices of a clique that is also a module. In other terms, we can assign two vertices the same set, provided that they are adjacent, and they have identical adjacencies with the remainder of the graph. This property of containment representations allows for compact representations of cliques, collections of universal vertices, and other subgraphs that contain many similar vertices, or any graph that can be build primarily using vertex expansion. We have the following lemma for containment representations, which is identical to Lemma 2.2 on intersection representations.

Lemma 4.2. *If H can be obtained from G by vertex expansion, then $\xi(G) = \xi(H)$.*

Proof. Since G is an induced subgraph of H $\xi(G) \leq \xi(H)$, by restricting a representation for H to the vertices of G.

In the other direction, let $\mathcal{C} = \{S_v : v \in V\}$ be a containment representation for $G = (V, E)$. Each vertex v of G is mapped by vertex expansion to a clique in H. Let C_v be the clique resulting from the expansion of vertex v, where $|C_v|$ may be one. Each vertex of H is in exactly one clique C_v. We form the representation given by the following collection, for each vertex u of H,

$$\mathcal{D} = \{S'_u = S_v : u \in C_v\}.$$

Each vertex in C_v is assigned the same set, S_v, and since $S_v \subseteq S_v$ this is a correct representation of these vertices. For any two vertices w and z of H, where $w \in C_u$ and $z \in C_v$, the sets $S'_w = S_u$ and $S'_z = S_v$ have a containment relationship if and only if u and v are adjacent in G. Thus $\mathcal{D}$ is a containment representation for H, which proves that $\xi(H) \leq \xi(G)$. □

From the definition, containment representations are quite similar to intersection and overlap representations. Like overlap representations, in building a containment representation we seem to be forced to make a choice for each non-edge. This choice is between representing the non-edge as disjoint or overlapping sets, and similar to the case of overlap representations, this appears to make the problem of finding a minimum containment representation much more difficult than finding a minimum intersection representation. As an example, notice that there are non-edges in Figure 4.1 represented in both of these ways.

An arbitrary graph does not, in general, have a containment representation. This is in contrast to the cases of intersection and overlap representations, where a representation can be found for an arbitrary graph. The restriction on the graphs that have containment representations comes from the fact that any containment representation of a graph necessarily produces a transitive orientation of the edges of a graph, since we can order the edges by set containment. This does not handle those vertices assigned the same set, but in this case we can simply treat these vertices as one vertex when finding a transitive orientation. Once such an orientation is found, we can decide an arbitrary total ordering of these vertices and

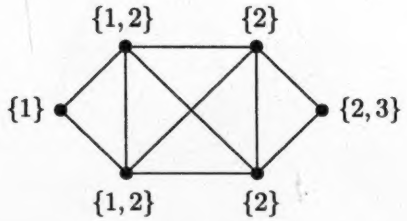

Figure 4.1: Example of a minimum containment representation.

replace the single vertex we had used in place of them, duplicating the edges attached to this single vertex. Since these vertices form both a clique and a module this process will not destroy the transitive orientation, and so we can construct a transitive orientation for any graph that has a containment representation. This forces such a graph to be a comparability graph.

This necessary condition is also sufficient, as noted in [27]. To see this, given any comparability graph, we choose an arbitrary transitive orientation, and assign each vertex v the set $S_v = \{u : (u, v) \in E\} \cup \{v\}$ of all vertices that are endpoints of edges directed toward v, plus the vertex v. To see that this forms a containment representation, let u, v be any two vertices, with S_u and S_v the sets associated with them. If $S_u \subseteq S_v$ then we have $u \in S_v$, which implies that (u, v) is a directed edge of the transitive orientation. If S_u and S_v are disjoint, then the vertices u and v must not be adjacent as $u \notin S_v$ and $v \notin S_u$. If, on the other hand, S_u and S_v overlap, but neither is contained in the other, and we assume for contradiction that u and v are adjacent, then there is some $x \in S_u$ with $x \neq u$ and $x \notin S_v$, and so, if (u, v) is a directed edge of the representation, we have (x, u) and (u, v) in the transitive orientation, but not (x, v), which is a contradiction that shows that (u, v) is not an edge of the graph, since we can reverse the roles of u and v to show that (v, u) is not an a directed edge in the orientation of the graph. Thus, this construction produces a valid containment representation for any comparability graph, and so the graphs that have containment representations are exactly the comparability graphs. It is also noted in [27] that the comparability graphs are exactly the graphs that can be represented as containment graphs of the subtrees of a tree, and this can be seen by taking any containment representation and arranging the elements as the leaves of a star. The sets of leaves are formed into subtrees by adding the root of the star to each set. Then, one of these subtrees will contain another if and only if one set of leaves contains the other, which happens exactly when one of the original sets in the containment representation contained the other.

There is little previous work on finding minimum containment representations. Representations of graphs by containment of restricted families of sets has been considered, but there has been little investigation into the case where the family of sets being considered are all subsets of $\{1, 2, \ldots, m\}$ for some m.

Golumbic and Scheinerman have also established necessary and sufficient conditions in [27] for a class of graphs to be a containment class of graphs of some family of sets. Any class of graphs that is hereditary, closed under vertex multiplication, and has a coherently transitively orientable composition sequence is a class of containment graphs of some family of sets. It is important to note, however, that in [27] two identical sets are not considered to have a containment relationship. Since here we do consider these sets to have a containment relationship, we can replace the requirement of closure under vertex multiplication with closure under vertex expansion. Having made this replacement, the only difference between this and the characterization of intersection graphs discussed in Section 2.1 is that the composition is required to be *coherently transitively orientable*, that is, the graphs of the sequence can be transitively oriented in such a way that for each graph G_i in the sequence of oriented graphs, the orientation on G_i agrees with the orientation on G_{i+1} for those vertices of G_{i+1} that are a part of the induced subgraph that forms a copy of the graph G_i.

The remainder of this chapter is organized as follows. Section 4.2 discusses some dimension notions on partially ordered sets that will be useful when examining containment representations of graphs. We examine hardness results related to the problem of finding a minimum containment representation in Section 4.3. In Section 4.4 we examine some bounds on the size of the containment number, and a connection to the overlap number, which we will use in Section 4.5 where we see some classes of graphs for which it is known how to efficiently construct a minimum containment representation.

4.2 Embedding a Partial Order into a Hypercube

A notion related to a containment representation of a graph is the notion of embedding a partial order into a hypercube. The elements of the partial order are associated with vertices of an m-dimensional hypercube. The vertices of this hypercube can be thought of as characteristic vectors of $\{1, 2, \ldots, m\}$, ordered by set containment. Since this yields a containment representation of the partial order, the task of minimizing the dimension of this hypercube thus seems to be related to the task of minimizing the size of a containment representation for a graph. As an example, consider the graph K_4. We can take any transitive orientation of this graph to obtain a partial order on four elements. Considering the transitive reduction of this partial order we have a directed path on four vertices, and so we seek the smallest hypercube into which we can embed a directed path of length three. Since any path of length three in a two dimensional hypercube, which is exactly a transitively oriented C_4, must use edges that are oriented in opposite directions, the smallest hypercube we can embed K_3 into is a three dimensional cube. The vertices of the hypercube we might embed the vertices of K_3 into could be $000, 001, 011$, and 111. The sets associated with these vertices are $\emptyset, \{1\}, \{1,2\}$, and $\{1,2,3\}$, and taking these sets forms a containment representation for the graph K_4. In Section 4.2.1 we will examine some notions from the theory of partially ordered sets, so that we can later apply some of them to the problem of finding the containment number of a graph. We conclude with Section 4.2.2, where we discuss some complexity issues related to computing the size of the smallest hypercube we can embed a partially ordered set into.

4.2.1 Partial Order Dimension Theory

In this section we introduce some definitions and results from the theory of partially ordered sets that will be useful when examining containment representations for graphs. This is by no means a complete introduction to this large area of study. For more background than is given here, see [60].

A partially ordered set, or *poset*, is given by a pair $(U, \leq)$ containing a set U of elements, and a partial order relation on U, given by $\leq$, which is a reflexive, antisymmetric, and transitive binary relation. All posets we will consider will be finite.

Given a partially ordered set $P = (U, \leq)$, we have the associated comparability graph $G = (U, E)$ on the elements of the poset, where $(u, v) \in E$ if and only if $u \leq v$ or $v \leq u$ in the partial order P. This graph is simply an undirected representation of P, and it is appropriate to call it the comparability graph of P, as the transitive orientation given by the partial ordering of P forms a transitive orientation of the vertices of G. In a similar way, given a comparability graph $G = (V, E)$, we may take any transitive orientation, and construct a poset $P = (V, \leq)$, where $u \leq v$ if and only if there is a directed edge from u to v in the chosen transitive orientation of G. Notice that a poset has a unique comparability graph, but since a comparability graph may have many possible transitive orientations, there may be many posets associated with a comparability graph. We call a property of a poset P *comparability invariant* if it is shared by all posets associated with the comparability graph of P.

A *chain* in a poset is a set of elements that are pairwise comparable. In other words, in the comparability graph of a poset, this set forms a clique. The height of a poset P, given by $h(P)$, is defined to be the size of the maximum clique in the comparability graph of the poset. This height of a poset is clearly comparability invariant, as any partial ordering of the vertices of a clique produces a chain of the same size.

A *linear extension* of a poset P is a totally ordered set Q on the same set of elements that is consistent with the partial ordering of P, that is, if $u \leq v$ in P then we have $u \leq v$ in the order given by Q as well. Using linear extensions we can build a representation of a poset. To represent a poset P, we take linear extensions $Q_1, Q_2, \ldots, Q_m$ such that for each pair of incomparable elements u and v in P there are Q_i and Q_j such that $u \leq v$ in Q_i and $v \leq u$ in Q_j. These linear extensions form a representation, since we can reconstruct P given only $Q_1, Q_2, \ldots, Q_m$, by noticing that for each pair of elements u, v, $u \leq v$ in each Q_i if and only if $u \leq v$ in P. The smallest m that allows such a representation is the *dimension* of the poset P, denoted $\dim(P)$. This notion of dimension was introduced in 1941 by Dushnik and Miller [14]. It is also known that the dimension of a partial order is a comparability invariant [58], and it is **NP**-complete to determine if the dimension of a poset is k, even for fixed $k \geq 3$ [66].

As an example of the dimension of a poset, consider the poset given by the divisors of the number 6. More precisely, the elements of this poset are $1, 2, 3, 6$, with $a \leq b$ if and only if a divides b. The dimension of this poset is two, since if we attempt to represent it with only one linear extension, we notice that we must have either $2 \leq 3$ or $3 \leq 2$ in the representation, but since neither of these is true in the poset under consideration, we need at least two linear extensions of the poset. If we use the two linear extensions given by $1 \leq 2 \leq 3 \leq 6$ and $1 \leq 3 \leq 2 \leq 6$, we obtain a representation of the poset, and so this poset has dimension two.

The definition of the dimension of a poset P of size n is concerned with embedding P into linear extensions, which form chains of size n. This embedding is then an embedding into the product of chains of length n. The *product* of two partial orders P and Q is an order of pairs in $P \times Q$, with $(a, c) \leq (b, d)$ in the product if and only if $a \leq b$ in P and $c \leq d$ in Q. A natural means of extending this definition is to bound the size of the chains that we embed the poset into. This is the approach taken by Trotter [59], who defines the k-dimension as the minimum number of chains of length k required to represent a poset P, denoted $\dim_k(P)$.

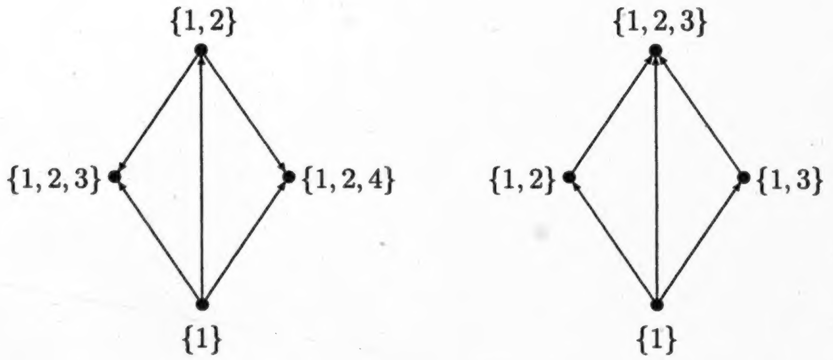

Figure 4.2: Two transitive orientations of the same graph with differing 2-dimension.

A special case of considerable interest is the 2-dimension of a poset, denoted $\dim_2(P)$. This case is also studied in [59], where it is observed that if a poset P is embedded into a product of m chains of length 2, then we can treat this embedding as an embedding into subsets of $\{1, 2, \ldots, m\}$, where if $u \leq v$ in P, then we have $S_u \subseteq S_v$, where S_u and S_v are the subsets associated with u and v by this embedding. The reason that the elements of products of chains of length two can be seen as sets is that in the product, for each element in the partial order, each of these chains corresponds to a bit, and so if we view these bits as characteristic vectors, we obtain an embedding into subsets of a set. Finding the 2-dimension is thus equivalent to finding the dimension of the smallest hypercube that a partial order can be embedded into.

The 2-dimension is thus of interest in the study of the size of a containment representation of a graph G, but there are obstacles to the direct computation of the containment number by the 2-dimension. The first of these obstacles is that the embedding into a product of chains given by the k-dimension is required to be injective, which forces different elements of the poset to be associated with different sets. This is a restriction that we have not made on containment representations, and there are cases when it can have a large effect. For example the 2-dimension of a clique on n vertices is n whereas the containment number of any clique is zero. The second, and perhaps more serious, obstacle is that the 2-dimension is not a comparability invariant. This is demonstrated by Figure 4.2, where the 2-dimension is given by providing an embedding into subsets of $\{1, 2, \ldots, m\}$ for minimum m. The containment number of a graph G, where we require that no two vertices are assigned the same set, is thus given by the minimum over all transitive orientations of the 2-dimension of the associated partially ordered set. Despite these obstacles, there are some special cases where we can apply results concerning the 2-dimension to the containment number of a graph.

We conclude with one such case. The following theorem of Trotter [59] that bounds the size of the 2-dimension for any partially ordered set, will be used to upper bound the containment number in Section 4.4.

Theorem 4.3 (Trotter [59]). *For any poset P with n elements,*

$$\log n \leq \dim_2(P) \leq n.$$

4.2.2 Complexity of Calculating the 2-Dimension

In this section we examine many of the known complexity results related to the 2-dimension of a partially ordered set. These results indicate that the 2-dimension is a difficult quantity to compute, and we will see that it is also difficult to approximate.

The **NP**-completeness of the problem of calculating the 2-dimension of a poset is briefly argued by Stahl and Wille [53]. A more thorough argument is given by Habib et al. [30], where the analysis is done carefully so as to show the following non-approximability result, which is quite a strong non-approximability result.

Theorem 4.4 (Habib, Nourine, Raynaud, and Thierry [30]). *For all $\epsilon > 0$, all $k \geq 2$, and all partial orders P, there is no polynomial time algorithm approximating $\dim_k(P)$ within $O(n^{1/21-\epsilon})$ unless* **P** = **NP**.

In addition to this theorem, Habib et al. show that it is computationally difficult to determine if the height of a poset is equal to its 2-dimension. The height forms a lower bound on the value of the 2-dimension, as a chain on n elements has 2-dimension n, and so, the following theorem shows that it is hard to determine, for an arbitrary poset, if this lower bound can be achieved.

Theorem 4.5 (Habib, Nourine, Raynaud, and Thierry [30]). *Deciding if $dim_2(P) = h(P)$ for any order P is* **NP***-complete.*

In addition to these hardness results, there are few classes of posets for which it is known how to, in polynomial time, compute the 2-dimension. Considerable effort [7, 8, 30, 35] has been applied to the case of partially ordered sets whose transitive reduction forms a tree, as this problem has applications in the representation of multiple inheritance hierarchies for object oriented programming languages. Despite this interest, the best performing algorithm, which is given in [30], is only able to approximate the 2-dimension of these posets within a factor of 4. This paper also contains the conjecture that the 2-dimension of these posets can be computed in polynomial time.

4.3 Hardness Results

While one might expect that the hardness of calculating the 2-dimension of a partially ordered set would easily imply the hardness of the problem of calculating the containment number of a graph this does not appear to be the case. The construction used in [30] to show the **NP**-completeness of the 2-dimension problem is heavily dependent upon the specific partial order, and it is not clear that finding the minimum 2-dimension over all transitive orientations of a graph is a difficult problem. The 2-dimension problem adds evidence that the problem of computing the containment number is **NP**-complete, but as in the case of the overlap number, the problem remains open. Once again, the problem of finding a minimum containment representation appears to be more involved than finding a minimum intersection representation, as we must make a nontrivial choice between disjointedness and overlap for each non-edge of the graph. For completeness, the following is a formalization of this problem.

Problem. The CONTAINMENT NUMBER problem is defined as:

Instance: A graph, $G = (V, E)$, and an integer k.

Question: Is there a containment representation $\mathcal{C} = \{S_v : v \in V\}$ of G such that

$$\left| \bigcup_{v \in V} S_v \right| \leq k?$$

We show the hardness of the problem of extending an overlap representation in the next section.

4.3.1 Extending a Containment Representation

We consider once more the problem of deciding if a representation can be extended to include an additional vertex without increasing the size of the representation. As in the case of the overlap representation, which we saw in Section 3.3.1, this problem is hard for containment representations, unlike the case for intersection representations for which we saw a polynomial-time algorithm in Section 2.3.5. In order to demonstrate the hardness of the problem, we consider the following formalized version of the problem of extending a containment representation.

Problem. The CONTAINMENT EXTENSION problem is defined as:

Instance: A graph, $G = (V, E)$, a containment representation $\mathcal{C} = \{S_v : v \in V\}$ of G, and a set $A \subseteq V$.

Question: Is there a set $S \subseteq \bigcup_{v \in V} S_v$ such that either $S \subseteq S_v$ or $S_v \subseteq S$ if and only if $v \in A$?

We can show that this problem is hard by reducing an instance of 3SAT to it. This will be done in a similar way to the proof of the hardness of STAR OVERLAP EXTENSION, where adding an element to the representation for the new vertex will have the effect of setting the truth value of a variable in the 3SAT instance. This reduction is presented in the proof of the next theorem.

Theorem 4.6. CONTAINMENT EXTENSION *is* **NP**-*complete.*

Proof. Once again, demonstrating that the problem is in **NP** is easy, as given a set S, we can check, in polynomial time, by considering each pair of vertices, and each element in the representation, whether S satisfies the requirements of the problem.

To show the **NP**-hardness of the problem, we will reduce a given instance of the **NP**-complete 3SAT problem to an instance of the CONTAINMENT EXTENSION problem. To this end, let (U, F) be an instance of 3SAT, where $U = \{x_1, x_2, \ldots, x_n\}$ is a set of n variables, and $F = \{c_1, c_2, \ldots, c_m\}$ is a set of m clauses, each containing three literals. We will restrict our attention to the case where $n \geq 4$, which can be done without loss of generality by simply solving any smaller instances and producing as output of the reduction a trivial yes or no instance of CONTAINMENT EXTENSION.

From (U, F), we construct a graph $G = (V, E)$ with containment representation $\mathcal{C} = \{S_v : v \in V\}$. The vertices of the graph are given by

$$V = \{v_i : 1 \leq i \leq n\} \cup \{w_i : 1 \leq i \leq m\} \cup \{z\},$$

where the vertices v_i will be associated with variables in U, the vertices w_i will be associated with clauses in F, and the vertex z is used to ensure that an extension of the representation forms a truth assignment to the variables of U. If we set $L = \bigcup_{i=1}^{n} \{x_i, \neg x_i\}$, the set of all literals, the containment representation that will be part of the instance of CONTAINMENT EXTENSION is given by the collection $\mathcal{C}$ consisting of the following sets, for all

$1 \leq i \leq n$ and $1 \leq j \leq m$,

$$\begin{aligned} S_{v_i} &= \{x_i, \neg x_i\} \\ S_{w_i} &= (U \setminus c_i) \cup \{0\} \\ S_z &= \{0\}. \end{aligned}$$

The sets, S_{v_i}, of the representation associated with the vertices v_i will be used to force an extension of the containment representation to choose only one truth value for each variable, while the sets S_{w_j}, associated with the w_j, will ensure that the choices made satisfy the clauses of the 3SAT instance. The remaining set of the representation $S_z = \{0\}$, which is associated with the vertex z, is used only to ensure that any set extending the containment representation must intersect the sets representing the clauses of the 3SAT instance, as this will force the constructed representation to satisfy these clauses. To complete the construction of the instance of CONTAINMENT EXTENSION, we set $A = \{z\}$, the set of neighbours of the vertex associated with the set S, and we set E, the edges of the graph to be those edges implied by the containment representation $\mathcal{C}$. This construction can clearly be performed in polynomial time.

To see that the reduction is correct, consider any truth assignment to the variables of U that satisfies the clauses of F. Consider the set S given by the literals that the truth assignment makes true, and the element 0. $S_z = \{0\} \subseteq S$, which satisfies the condition that the vertex associated with S is adjacent to z. Also, for any variable x_i, S contains exactly one of x_i and $\neg x_i$, which implies that $S_{v_i} \not\subseteq S$, and in addition, since $n > 1$, there are at least two variables, so that $S \not\subseteq S_{v_i}$. Thus the sets associated with the vertices v_i have the desired relationship with the set S. If we take any $S_{w_i} = (L \setminus c_i) \cup \{0\}$, there is some literal in c_i that is made true by the truth assignment, and this literal does not appear in S_{w_i}, so we have $S \not\subseteq S_{w_i}$. In addition, since we have imposed the restriction that $n \geq 4$, and c_i contains exactly three literals, there is some variable x_j such that x_j and $\neg x_j$ both appear in S_{w_i}, but if both of these variables are in S, then $S_{v_j} \subseteq S$, which we have already shown is not the case. Thus $S_{w_i} \not\subseteq S$, and the set S has the properties required by the CONTAINMENT EXTENSION problem.

In the other direction, we consider any set, S, that is a yes instance of CONTAINMENT EXTENSION. Since S forms a containment relationship with $S_z = \{0\}$, we must have $0 \in S$, as otherwise we have $S \subseteq \{0\} \subseteq S_{w_i}$ for any i, which contradicts the choice of S, as the set S does not have a containment relationship with S_{w_i} for any i. Since we have $0 \in S$, the set S intersects S_{w_i} for all i, and since there is no containment, there must be an element in each set that is not in the other. The only elements of the representation not in S_{w_i}, for each i, are the literals that satisfy the clause c_i, and so there are literals in S that satisfy each clause. Also, since $S_{v_i} = \{x_i, \neg x_i\} \not\subseteq S$, for all i, there is no variable in S that appears in both plan and negated form. Thus, we can choose a partial truth assignment that makes true all the literals of S, and this assignment will also satisfy the instance of 3SAT. This assignment may not assign truth values to all variables, but this is easily remedied by assigning arbitrary truth values to the remaining variables. We have thus shown that the instance of 3SAT is satisfiable if and only if there is a set S that satisfies the requirements of the constructed instance of CONTAINMENT EXTENSION. Since this problem is contained in **NP**, as argued above, we have shown that CONTAINMENT EXTENSION is **NP**-complete. $\square$

4.4 Bounds on the Size of a Containment Representation

In Section 4.2.1 we have seen both upper and lower bounds on the size of the 2-dimension for any partially ordered set. If we consider the 2-dimension problem to be the problem of embedding a partial order into a collection of sets ordered by containment, there is still one critical difference between this problem and the problem of finding a containment representation. This difference is that in embedding a partial order into a collection of sets for the 2-dimension problem, we only consider embeddings that are injective, which is, each vertex of the partial order is assigned a distinct set. In a containment representation we do not make this restriction, and this will cause us to lose the lower bound of Theorem 4.3, which can be verified by noting that the containment number of a clique is zero. We are still able to extend the upper bound of this theorem to containment representations. This is given as the following corollary.

Corollary 4.7. *If G is a comparability graph with n vertices, then $\xi(G) \leq n$.*

Proof. Let P be the partially ordered set formed by any transitive orientation of G. Notice that any embedding of P into a hypercube will also form a containment representation for G. Thus, by Theorem 4.3, we have $\xi(G) \leq \dim_2(P) \leq n$. $\square$

This bound is tight for small values of n, as an independent set of size two has containment number two, and the bound used on the two dimension is tight, as a chain on n vertices, which is a clique in the language of graph theory, has 2-dimension n. It is not known, however, if this tightness is dependent on the fact that the embeddings of partial orders into hypercubes are required to be injective, and since we have not made this restriction in the case of a containment representation, it is not known if the bound in the above corollary is tight for all values of n.

This bound is in contrast to the quadratic bound given on the intersection number in Theorem 2.16, since the example graph that achieves the quadratic bound is a comparability graph. This is similar to the bound for overlap representations given as Theorem 3.31, and we will study a relationship between containment and overlap representations in the next section.

4.4.1 Relation to Overlap Representations

In this section we examine the relationship between the size of a minimum overlap representation for a graph, and the size of a minimum containment representation for the complement. On the surface, it is a surprise that these two quantities have any relationship at all, given that there does not seem to be a straightforward method to convert overlap representations to containment representations.

In order to study this relationship between overlap and containment representations, it is helpful to examine a restriction of the problem of finding a general representation. The representations we seek must satisfy the additional property of having no disjoint sets. We introduce notation for these concepts, letting $\varphi'(G)$ and $\xi'(G)$ be the minimum size of disjointedness-free overlap and containment representations of the graph G. This restriction, as we show in the next lemma, forces any containment representation to also be an overlap representation of the complement graph. This lemma also implies that the minimum disjointness-free overlap representation for $\overline{G}$ has the same size as the minimum disjointness-free containment representation for G.

Lemma 4.8. *Let* $G = (V, E)$ *be a comparability graph. The collection* $\mathcal{C} = \{S_v : v \in V\}$ *is a containment representation for G with no disjoint sets if and only if* $\mathcal{C}$ *also forms an overlap representation for* $\overline{G}$ *with no disjoint sets.*

Proof. Consider $\mathcal{C}$ a disjointedness-free set representation for a graph with vertices V. The collection $\mathcal{C}$ is a containment representation for G if and only if $(u, v) \in E$ implies $S_u \subseteq S_v$ or $S_v \subseteq S_u$, and $(u, v) \notin E$ implies that S_u and S_v overlap. This occurs exactly when $\mathcal{C}$ forms an overlap representation for $\overline{G}$. □

With this lemma, we can prove a relationship between the overlap and containment numbers without the disjointedness-free restriction. The proof is a simple application of this lemma and an observation about containment representations.

Theorem 4.9. *For any comparability graph G,*

$$\varphi(\overline{G}) \leq \xi(G) + 1$$

Proof. Lemma 4.8 proves that for any comparability graph G, we have $\varphi'(\overline{G}) = \xi'(G)$. Since a disjointedness-free overlap representation is an overlap representation, we have

$$\varphi(\overline{G}) \leq \varphi'(\overline{G}) = \xi'(G). \tag{4.1}$$

We then notice that from any containment representation, we can form a disjointedness-free representation by adding a single new element to all sets of the representation. Thus, we know that $\xi'(G) \leq \xi(G) + 1$, and combining this with Equation (4.1) yields the desired bound. ☐

This theorem, with the result of Theorem 4.7, provides a different proof of Theorem 3.31, but it can also be used to take an upper bound on the size of a containment representation for any subclass of comparability graphs to a corresponding upper bound for an overlap representation on the complementary class of graphs.

4.5 Algorithms

There are few classes of graphs that have known algorithms to find minimum containment representations. This is perhaps not a surprise when we consider the scarcity of algorithms to compute the 2-dimension of a partial order, as we have seen in Section 4.2.2.

Despite this scarcity, we will present algorithms to find minimum containment representations for a few classes of graphs. These results are primarily translations of some results on the overlap number to the containment number for the complement classes of graphs. In Section 4.5.1 we will see a scheme to find minimum containment representations of independent sets, and in Section 4.5.2, we will examine containment representations for the complements of paths and caterpillars.

4.5.1 Independent Sets

One small class of graphs for which we can find a minimum containment representation is the class of independent sets. While these graphs are rather trivial, we still require the application of deep combinatorial results to find the minimum size of such a representation.

The maximum number of subsets of $\{1, 2, \ldots, m\}$ in a collection such that no set in the collection contains any other is given by Sperner's Theorem [51], which is given here as the case where $p = 0$ of Theorem 3.4. As we are considering independent sets, no two vertices may be assigned the same set, since this implies a containment relationship. For this reason, the number of elements needed to form a containment representation for n independent vertices is given by the quantity in Sperner's Theorem. This quantity can be at most one smaller, and no larger, than the overlap number for a clique on n vertices, which can be observed either from the structure of the function $S(p, m)$, or Table 3.1. If we apply Theorem 4.9 to the $p = 1$ case of Corollary 3.5, we obtain a bound that is never more than one higher than the bound given by the following corollary.

Corollary 4.10. *A minimum containment representation for an independent set on n vertices has size given by*

$$\min \{m : n \leq S(0, m)\}$$

Proof. By Theorem 3.4 $S(0, m) = \binom{m}{\lfloor (m+1)/2 \rfloor}$ is the maximum number of subsets of $\{1, 2, \ldots, m\}$ such that no set contains any other, and so, where m is the minimum value satisfying the minimum in the theorem, a containment representation must use at least m elements.

To construct such a representation, we take the value m from the theorem, and consider all subsets of size $\lfloor (m + 1)/2 \rfloor$. Since these subsets all have the same size, none will be contained in any other, and counting them we find that there are

$$\binom{m}{\lfloor \frac{m+1}{2} \rfloor} = S(0, m) \geq n$$

such sets, and so we may form a containment relationship for an independent set on n vertices by choosing any n of them. ☐

In order to actually find such sets, we can use the results in Section 3.2.3 to compute the minimum value m such that $n \leq S(0, m)$ in linear time. From this value, we simply produce n arbitrary subsets of $\{1, 2, \ldots, m\}$, which can be done in time exponential in m; since $m \in O(\log n)$, by Theorem 3.10, this can be performed in time linear in the size of the graph. Thus we have a linear algorithm to find a minimum containment representation of an independent set.

4.5.2 Complements of Paths and Caterpillars

In Section 4.4.1 we have seen a relation between the overlap number of a graph and the containment number of the complement. Specifically, Theorem 4.9 enables us to translate upper bounds on the overlap number to lower bounds on the containment number. In this section we will make use of it to show some tight lower bounds on the complements of paths and caterpillars. These bounds will be shown tight by providing a scheme for constructing a minimum containment representation for one of these graphs. The first of these results enables us to construct a minimum containment representation for the complement of a path on n vertices.

Corollary 4.11. *For* $n \geq 2$, *we have* $\xi(\overline{P_n}) = n - 1$.

Proof. By Theorem 3.15, $\varphi(P_n) = n$ for $n \geq 3$. Theorem 4.9 lets us transfer this to a lower bound on a containment representation of $\overline{P_n}$, so we have $\xi(\overline{P_n}) \geq n - 1$, and so it is sufficient to construct a containment representation of the correct size. To observe the desired lower bound on the $\overline{P_2}$, notice that we require two sets such that neither contains the other, which requires that they each have a unique element.

To construct a representation that achieves this lower bound, we let $\{1, 2, \ldots, n\}$ be the vertices of $\overline{P_n}$. Consider the sets given by, for $3 \leq i < n$,

$$\begin{aligned} S_1 &= \{1\} \\ S_2 &= \{2\} \\ S_i &= \{1, 2, \ldots, i-2\} \cup \{i\} \\ S_n &= \{1, 2, \ldots, n-2\}. \end{aligned}$$

This representation, by construction, has size $n - 1$. It can also be transformed into an overlap representation for P_n if the element n is added to each set. Notice that for any i and j, we have $S_i \subseteq S_j$ if and only if $i < j - 1$, as the set S_i contains only elements in $\{1, 2, \ldots, i\}$, while for any $j \geq i + 2$, the set S_j contains the set $\{1, 2, \ldots, j-2\} \supseteq \{1, 2, \ldots, i\}$. Thus, if for each i we associate the set S_i with the vertex i in the containment representation given by $C = \{S_i\}$, vertices i and j are adjacent if and only if they are at distance at least two in P_n, which is exactly the adjacency condition in $\overline{P_n}$. □

Since this proof is constructive, we can use the obvious algorithm to construct a minimum containment representation of $\overline{P_n}$ in $O(n^2)$ time. The size of the constructed representation, in terms of the sum of the number of elements in each set, in this case is required to be quadratic, as all containments in the representation must be proper if we have $n > 3$. Since the containments are proper, and we can find a chain of $\lceil n/2 \rceil$ of them, we require at least $\sum_{i=1}^{\lceil n/2 \rceil} i \in \Omega(n^2)$ elements in the sum of the sizes of the sets representing this chain. Thus, this algorithm is optimal in both asymptotic run time and the size of the constructed representation.

Having extended the overlap number of a path to the containment number of the complement of a path, we can also extend in a similar way the bounds on the overlap number of a caterpillar, which, as given in Definition 3.17, is a tree with the restriction that all non-leaf vertices form a path, called the *spine* of the caterpillar. In determining the containment number of the complement of a caterpillar, we can use as a lower bound either the above bound on the containment number of the complement of the spine, or we can once again use Theorem 4.9 to obtain a bound from the value of the overlap number for the caterpillar. The containment number of a caterpillar will be one smaller than the overlap number of the complement, as we can once again remove a single element from an overlap representation.

Corollary 4.12. *For $T = (V, E)$ a caterpillar with spine containing $k \geq 1$ vertices, we have $\xi(\overline{T}) = k + 1$.*

Proof. By Corollary 3.18, which is the statement for overlap representations of caterpillars, and Theorem 4.9 we know that $\xi(\overline{T}) \geq k+1$, and so we need only construct a representation that matches this bound.

In order to construct such a representation, we let T be a caterpillar, with v_i for $1 \leq i \leq k$ the vertices of the spine in order, and we let L_i be the set of all leaves adjacent to v_i. Consider the representation given by

$$\begin{aligned} S_{v_i} &= \{1, 2, \ldots, i-1\} \cup \{i+1\} \\ S_{L_i} &= \{1, 2, \ldots, i\}, \end{aligned}$$

where we assign to each vertex v_i the set S_{v_i} and to each vertex in L_i we assign the set S_{L_i}. Once again, this forms an overlap representation for T if we add the element $k+2$ to each set. To see that this containment representation for $\overline{T}$ is correct, notice that for any $i \leq j$ we have $S_{L_i} \subseteq S_{L_j}$, and this is the correct relationship between the leaves of the caterpillar, as in the complement these leaves are all adjacent. If we consider the relationship between the leaves and the vertices on the spine, notice that if $i < j$ we have $S_{v_i} \subseteq S_{L_j}$ and $S_{L_i} \subseteq S_{v_j}$, which correctly represents the edges between any leaf and vertex of the spine it is not adjacent to in T. We also have, for any i, an overlap between the sets S_{v_i} and S_{L_i} as the elements $i+1$ and i are unique to these sets, respectively. Thus in the complement of the caterpillar, a leaf will correctly not be adjacent to the corresponding vertex on the spine. Finally, we consider the case of any two vertices on the spine; since this representation is identical to the one used in the proof of Corollary 4.11, we know that $S_{v_i} \subseteq S_{v_j}$ if and only if $i < j-1$, and this correctly represents the complement of the spine of the caterpillar. □

Once again, this representation takes quadratic space to specify, as the sum of sizes of the constructed sets is quadratic in k, the length of the spine of the caterpillar. Thus, the simple algorithm that produces this representation runs in time $O(k^2)$, which is again asymptotically optimal as we require at least $\Omega(k^2)$ space to specify a containment representation for just the vertices forming the complement of the spine of the caterpillar.

Chapter 5

Conclusions and Open Problems

We have seen that the study of the sizes of set representation for graphs contains many interesting results and has deep ties to elegant areas of mathematics. These ideas are also useful in the broader context of representations of graphs, which makes this an important area of study.

In addition to the problems we have considered, there are problems on set representations that remain unsolved. In this section we state some of the open problems related to the results presented here. The primary open problem is formulated as the following conjecture.

Conjecture 5.1. OVERLAP NUMBER, the problem of determining if an input graph has an overlap representation of a given size, is **NP**-complete.

This problem is clearly in **NP**, as it is a simple matter to verify that a given representation is both correct and of the appropriate size. The justification for conjecturing that the problem is **NP**-complete is that many problems on overlap representations appear to be harder than the related problems on intersection representations, and the INTERSECTION NUMBER is known to be **NP**-complete. Intersection representations are equivalent to edge-clique covers, but no similar equivalence is known for overlap representations where each element in the representation forms not a clique, but a cocomparability graph. The problem of extending an overlap representation is also **NP**-complete, but the same problem on intersection representations admits an efficient algorithm.

These reasons also apply to containment representations, where the extension problem is also **NP**-complete, and elements of the representation form comparability graphs and not cliques. In this case, however, a similar conjecture may not be appropriate. The reason for this is that for a graph to have a containment representation it must be a comparability graph, and so there is some hope that since we consider only a restricted class of graphs the following open problem may be resolved by an efficient algorithm.

Open Problem 5.2. What is the complexity of CONTAINMENT NUMBER, the problem of determining if an input graph has a representation of a given size?

From these more general problems, we can consider the problem of finding a minimum representation for a restricted class of graphs. One such class of graphs is the cographs, for which no efficient algorithm is known to find the intersection number. This is unusual, as cographs have a simple recursive structure under the modular decomposition that allows for a simple vertex clique covering algorithm, but it is not known how to extend this to the case of an edge-clique covering algorithm. Cographs are also a relatively small class of graphs, and since they contain no induced P_4, they are almost chordal. The intersection number problem is solvable on the chordal graphs, and the cographs that are not chordal are only those that contain C_4 as an induced subgraph. This implies that there is a relatively small class of graphs that an algorithm for finding the intersection number of a cograph needs to consider. Despite this, no such algorithm is known, and the technique of clique covering the edge-clique graph, used in the chordal case, does not seem to extend to cographs. This problem is given as the following open problem.

Open Problem 5.3. Is there an efficient algorithm to find a minimum intersection representation for any cograph?

The last two open problems we present are the problems of finding minimum overlap and containment representations for the class of trees. Despite the apparent simplicity of this class, no algorithms are known to find the overlap and containment numbers of graphs in this class. An algorithm is known for the overlap number of caterpillars, but it does not appear to extend to the class of trees.

Open Problem 5.4. Is there an efficient algorithm to find a minimum overlap representation or a minimum containment representation for any tree?

From this list of open problems it can be seen that the study of these set representations for graphs is far from over. There are many more interesting and relevant problems that have yet to be solved.

Bibliography

[1] Michael O. Albertson and Karen L. Collins. Duality and perfection for edges in cliques. *Journal of Combinatorial Theory, Series B*, 36(3):298–309, 1984.

[2] Egon Balas and Chang Sung Yu. On graphs with polynomially solvable maximum-weight clique problem. *Networks*, 19(2):247–253, 1989.

[3] Claude Berge. *Graphs and Hypergraphs*. Number 6 in North-Holland Mathematical Library. North-Holland, 1973.

[4] Béla Bollobás. *Modern Graph Theory*. Springer, 1998.

[5] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinard. *Graph Classes: A Survey*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 1999.

[6] Tiziana Calamoneri and Rossella Petreschi. Edge-clique graphs and the λ-coloring problem. *Journal of the Brazillian Computer Society*, 7(3):38–47, 2001.

[7] Yves Caseau. Efficient handling of mulitple inheritance hierarchies. In *Proceedings of the 8th Annual Conference on Object-oriented Programming Systems, Languages, and Applications*, pages 271–287, 1993.

[8] Yves Caseau, Michel Habib, Lhouari Nourine, and Oliver Raynaud. Encoding of multiple inheritance hierarchies and partial orders. *Computational Intelligence*, 15(1):50–62, 1999.

[9] Eowyn Čenek and Lorna Stewart. Maximum independent set and maximum clique algorithms for overlap graphs. *Discrete Applied Mathematics*, 131(1):77–91, 2003.

[10] Márcia R. Cerioli. A characterization of edge clique graphs. *Ars Combinatoria*, 60:287–292, 2001.

[11] Márcia R. Cerioli and Jayme L. Szwarcfiter. Edge clique graphs of some classes of chordal graphs. *Discrete Mathematics*, 242(1–3):31–39, 2002.

[12] Gary Chartrand, S. F. Kapoor, Terry A. McKee, and Farrokh Saba. Edge-clique graphs. *Graphs and Combinatorics*, 7(3):253–264, 1991.

[13] Pierre Duchet. Classical perfect graphs: An introduction with emphasis on triangulated and interval graphs. *Annals of Discrete Mathematics*, 21:67–96, 1984.

[14] Ben Dushnik and E. W. Miller. Partially ordered sets. *American Journal of Mathematics*, 63(3):600–610, 1941.

[15] Nancy Eaton, Ronald J. Gould, and Vojtech Rödl. On p-intersection representations. *J. Graph Theory*, 21(4):377–392, 1996.

[16] Paul Erdős, A. W. Goodman, and Louis Pósa. The representation of a graph by set intersections. *Canadian Journal of Mathematics*, 18:106–112, 1966.

[17] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–855, 1965.

[18] Zoltán Füredi. Intersection representations of the complete bipartite graph. *Algorithms and Combinatorics*, 14:86–92, 1997.

[19] M. R. Garey, D. S. Johnson, and L. Sotckmeyer. Some simplified NP-complete problems. In *Proceedings of the 6th Annual ACM Symposium on Theory of Computing*, pages 47–63, 1974.

[20] M. R. Garey, D. S. Johnson, and R. Endre Tarjan. The planar Hamiltonian circuit problem is NP-complete. *SIAM Journal on Computing*, 5(4):704–714, 1976.

[21] Michael Garey and David Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[22] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16:47–56, 1974.

[23] Fanica Gavril. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3:261–273, 1973.

[24] Fanica Gavril. Maximum weight independent sets and cliques in intersection graphs of filaments. *Information Processing Letters*, 73:181–188, 2000.

[25] Fănică Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1(2):180–187, 1972.

[26] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Number 57 in Annals of Discrete Mathematics. Elsevier, second edition, 2004.

[27] Martin Charles Golumbic and Edward R. Scheinerman. *Containment Graphs, Posets, and Related Classes of Graphs*, pages 192–204. Number 555 in Annals of the New York Academy of Sciences. New York Academy of Sciences, 1989.

[28] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsiod method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[29] U. I. Gupta, D. T. Lee, and J. Y.-T. Leung. Efficient algorithms for interval graphs and circular-arc graphs. *Networks*, 12(4):459–467, 1982.

[30] M. Habib, L. Nourine, O. Raynaud, and E. Thierry. Computational aspects of the 2-dimension of partially ordered sets. *Theoretical Computer Science*, 312(2-3):401–431, 2004.

[31] David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.

[32] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[33] Donald E. Knuth. *The Art of Computer Programming*, volume 1, Fundamental Algorithms. Addison-Wesley, third edition, 1997.

[34] L. T. Kou, L. J. Stockmeyer, and C. K. Wong. Covering edges by cliques with regard to keyword conflicts and intersection graphs. *Communications of the Association for Computing Machinery*, 21(2):135–139, 1978.

[35] Andreas Krall, Jan Vitek, and R. Nigel Horspool. Near optimal hierarchical encoding of types. In *Proceedings of the 11th European Conference on Object Oriented Progamming*, pages 128–145, 1997.

[36] K. Kuratowski. Sur le probléme des courbes gauches en topologie. *Fundamenta Mathematicae*, 15:271–283, 1930.

[37] D. T. Lee and Joseph Y.-T. Leung. On the 2-dimensional channel assignment problem. *IEEE Transactions on Computers*, 33(1):2–6, 1984.

[38] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th IEEE Symposium on Switching and Automata Theory*, pages 125–129, 1972.

[39] E. C. Milner. A combinatorial theorem on systems of finite sets. *Journal of the London Mathematical Society*, 43:204–206, 1966.

[40] J. W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3:23–28, 1965.

[41] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[42] Svatopluk Poljak, Vojtěch Rödl, and Daniel Turzík. Complexity of representation of graphs by set systems. *Discrete Applied Mathematics*, 3(4):301–312, 1981.

[43] Erich Prisner. Graphs with few cliques. In *Proceedings of the 7th Quadrennial International Conference on the Theory and Applications of Graphs, 1992*, Graph Theory, Combinatorics, and Applications, pages 945–956, 1995.

[44] Erich Prisner. Generalized octahedra and cliques in intersection graphs of uniform hypergraphs. *Discrete Mathematics*, 206:187–195, 1999.

[45] Arundhati Raychaudhuri. Intersection number and edge clique graphs of chordal and strongly chordal graphs. *Congressus Numeratum*, 67:197–204, 1988.

[46] Arundhati Raychaudhuri. Edge clique graphs of some important classes of graphs. *Ars Combinatorica*, 32:269–278, 1991.

[47] Chong S. Rim and Kazuo Nakajima. On rectangle intersection and overlap graphs. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 42(9):549–553, 1995.

[48] Fred S. Roberts. On the boxicity and cubicity of a graph. In W. T. Tutte, editor, *Recent Progress in Combinatorics*, pages 301–310. Academic Press, 1969.

[49] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 216–226, 1978.

[50] Edward R. Scheinerman. Characterizing intersection classes of graphs. *Discrete Mathematics*, 55(2):185–193, 1985.

[51] E. Sperner. Ein satz über untermengen einer endlichen menge. *Mathematische Zeitschrift*, 27:544–548, 1928.

[52] Jeremy P. Spinrad. *Efficient Graph Representations*. Number 19 in Fields Institute Monographs. American Mathematical Society, 2003.

[53] Jürgen Stahl and Rudolf Wille. Preconcepts and set reprentation of contexts. In W. Gaul and M. Schader, editors, *Classification as a Tool of Research*. North-Holland, 1986.

[54] L. J. Stockmeyer. The polynomial hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[55] Larry Stockmeyer. Planar 3-colorability is polynomial complete. *SIGACT News*, 5(3):19–25, 1973.

[56] Edward Szpilrajn-Marczewski. Sur deux propriétés des classes d'ensembles. *Fundamenta Mathematicae*, 33:303–307, 1945.

[57] Seinosuke Toda. On the computational power of PP and $\oplus$P. In *Proceedings of the 30th IEEE Symposium on the Foundations of Computer Science*, pages 514–519, 1989.

[58] W. T. Trotter, John I. Moore Jr., and David P. Sumner. The dimension of a comparability graph. In *Proceedings of the American Mathematical Society*, volume 60, pages 35–38, 1976.

[59] William T. Trotter. Embedding finite posets in cubes. *Discrete Mathematics*, 12(2):165–172, 1975.

[60] William T. Trotter. *Combinatorics and Partially Ordered Sets: Dimension Theory*. Johns Hopkins University Press, 1992.

[61] Shuji Tsukiyama, Mikio Ide, Hiromu Arioyoshi, and Isao Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 6(3):505–517, 1977.

[62] Salil P. Vadhan. The complexity of counting in sparse, regular, and planar graphs. *SIAM Journal on Computing*, 31(2):398–427, 2001.

[63] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[64] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, second edition, 2001.

[65] H. S. Witsenhausen. On intersection of interval graphs. *Discrete Mathematics*, 31(2):211–216, 1980.

[66] Mihalis Yannakakis. The complexity of the partial order dimension problem. *SIAM Journal on Algebraic and Discrete Methods*, 3(3):351–358, 1982.

Appendix A

NP-complete Problems

In this appendix we give the definitions of some **NP**-complete problems that were not included in Section 2.5. These problems can be found in the book by Garey and Johnson [21].

The first of these problems is one that was one of the first problems known to be **NP**-complete. It is the 3SAT problem, shown to be **NP**-complete by Karp [32]. A definition of this problem follows.

Problem. The 3SAT problem is defined as:

Instance: A set U of variables, and a collection F of clauses over these variables, each containing exactly three literals.

Question: Is there is some truth assignment to the variables in U that satisfies all clauses in F?

A variant of this problem, which is also known to be **NP**-complete [49], is a restricted version of 3SAT where we require not only a satisfying truth assignment, but one that does not make all the literals in any clause true.

Problem. The NOT-ALL-EQUAL 3SAT problem is defined as:

Instance: A set U of variables, and a collection F of clauses over these variables, each containing exactly three literals.

Question: Is there is some truth assignment to the variables in U such that there is no clause of F with all three literals assigned the same truth value?

The next problem we introduce is also among those shown to be **NP**-complete in [32]. This is the problem of determining if a graph contains a clique of a given size.

Problem. The CLIQUE problem is defined as:

Instance: A graph, $G = (V, E)$, and an integer k.

Question: Is K_k is a subgraph of G?

The weighted version of CLIQUE is also **NP**-complete. This is because setting all weights to one results in the CLIQUE problem. This problem is as follows.

Problem. The WEIGHTED CLIQUE problem is defined as:

Instance: A graph, $G = (V, E)$, a function $w : V \to \mathbb{Z}$, and an integer k.

Question: Is there is a set of vertices S that induce a complete subgraph G such that

$$\sum_{v \in S} w(v) \geq k?$$

The CLIQUE problem is clearly equivalent to finding an independent set of the same size. The reduction is simply to complement the graph being considered. Thus, this problem is also **NP**-complete.

Problem. The INDEPENDENT SET problem is defined as:

Instance: A graph, $G = (V, E)$, and an integer k.

Question: Is there is a set $S \subseteq V$ such that $|S| = k$ and S induces a subgraph of G that is an independent set?

The next problem we introduce is another famous problem. This is the problem of determining the minimum number of colours needed to colour a graph such that no two adjacent vertices are assigned the same colour. This problem is also shown to be **NP**-complete in [32].

Problem. The COLOURABILITY problem is defined as:

Instance: A graph, $G = (V, E)$, and an integer k.

Question: Is there a function $c : V \to \{1, 2, \ldots, k\}$ such that for any $(u, v) \in E$ the function satisfies $c(u) \neq c(v)$?

The next problem we introduce is COLOURABILITY with a fixed constant value of k. This restriction of the problem is still **NP**-complete, as shown by Stockmeyer in [55].

Problem. The k-COLOURABILITY problem is defined as:

Instance: A graph, $G = (V, E)$.

Question: Is there a function $c : V \to \{1, 2, \ldots, k\}$ such that for any $(u, v) \in E$ the function satisfies $c(u) \neq c(v)$?

The problem of choosing the minimum set of vertices in a graph such that each edge is incident to some vertex of the set is also known to be **NP**-complete [32]. A formalization of this problem is the VERTEX COVER problem.

Problem. The VERTEX COVER problem is defined as:

Instance: A graph, $G = (V, E)$, and an integer k.

Question: Is there is a set $S \subseteq V$ such that $|S| = k$ and for each $(u, v) \in E$, at least one of u and v is in S?

The problem, similar to VERTEX COVER, of finding a set S of vertices such that all vertices not in the set are adjacent to a vertex in S. This problem is shown to be **NP**-complete in [21]. This problem is formalized as the DOMINATING SET problem.

Problem. The DOMINATING SET problem is defined as:

Instance: A graph, $G = (V, E)$, and an integer k.

Question: Is there is a set $S \subseteq V$ such that $|S| = k$ and for each $v \in V \setminus S$ there is some $u \in S$ with $(u, v) \in E$?

The final two problems that we introduce are the problems of finding a Hamiltonian path or a Hamiltonian cycle, where a path or cycle is Hamiltonian if it visits all the vertices of the graph, without repeating any vertices. The HAMILTONIAN CYCLE problem is shown to be **NP**-complete in [32].

Problem. The HAMILTONIAN CYCLE problem is defined as:

Instance: A graph, $G = (V, E)$.

Question: Is there is an cycle in the graph that visits each vertex exactly once?

The HAMILTONIAN PATH problem is not shown to be **NP**-complete in [32], although this fact is generally attributed to this paper. A hardness proof for a restricted version of the problem appears in [19].

Problem. The HAMILTONIAN PATH problem is defined as:

Instance: A graph, $G = (V, E)$.

Question: Is there is a path in the graph that visits each vertex exactly once?